The Magazine For ALL Commodore Computer Users

TPUG Magazine

COMPUTER LANGUAGES

Structured Programming
An end to spaghetti

C-64 Pascal Compilers
Which one is best?

1571 Burst Mode
Start of New Series

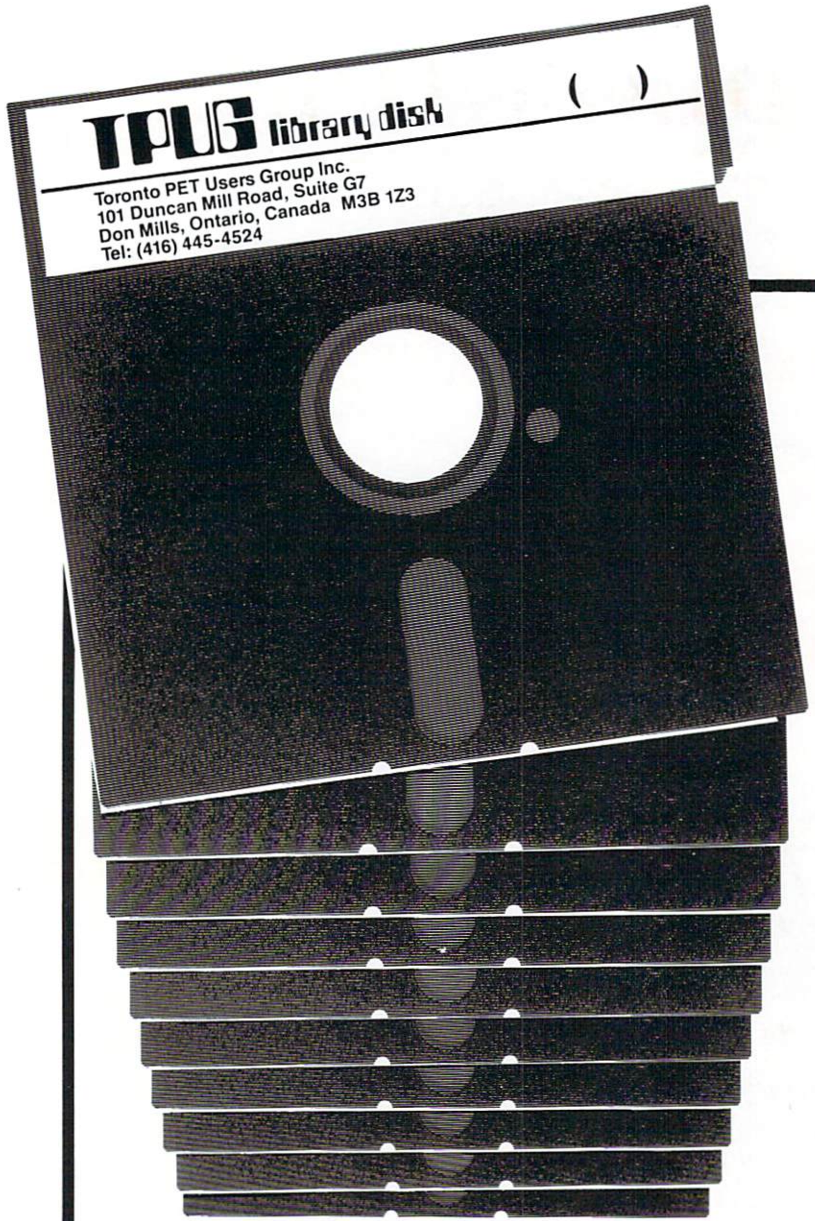
$2.95

Issue No. 23

779 A Oct 1986 4
Fort Wayne Area Commodore Club
P.O. Box 13107
Fort Wayne
IN
46867

Bulk rate
U.S. postage paid
Buffalo, N.Y.
Permit #1575

0 56698 70976 23


TPUG library disk ( )
Toronto PET Users Group Inc.
101 Duncan Mill Road, Suite G7
Don Mills, Ontario, Canada M3B 1Z3
Tel: (416) 445-4524

TPUG LIBRARY DISKS

10 months of the latest, fabulous public domain software only

$79.95 CANADIAN
US orders US$59.95

Subscribe now to start receiving the TPUG disk of the month

You **SAVE 20%** or more by using this offer as a TPUG member

THIS OFFER VALID FOR PAID UP TPUG MEMBERS ONLY
PLEASE PRINT

Name: ______________________ Membership # __________

Address: ______________________

____

Type of Payment

☐ Cheque
☐ Mastercard
☐ Visa
☐ Money Order

Amount $ __________
Currency ☐ Can. ☐ US
Credit Card # __________
Expiry Date __________

Signature __________

Type of Computer

☐ C 64
☐ VIC 20
☐ PET ☐ 4040 ☐ 8050
☐ SUPERPET (10 disks) ☐ 4040 ☐ 8050
☐ C128 (1541 only)
☐ MS/DOS
☐ AMIGA (3½") (7 disks) ☐

TPUG Magazine

Publisher: Bruce Hampson
Editor: Nick Sullivan
Assistant Editors: Tim Grantham
Adam Herst
Production Manager: Astrid Kumas
Editorial/Production Assistant: Iwona Sukiennik
Advertising Sales: John Matheson
Cover Illustration: Thom K. Wu
Creative Direction (Cover): Steve MacDowall
Cover Scan and Assembly: LK Graphics
Typesetting: Noesis, Toronto
Printed in Canada by: Delta Web Graphics
Scarborough, Ontario

TPUG Magazine is published 10 times a year by TPUG Inc. All rights to material published in TPUG Magazine are reserved by TPUG Inc., and no material may be reprinted without written permission except where specifically stated.

Correspondence: Send change of address and subscription enquiries to: TPUG Inc., Address Changes, 101 Duncan Mill Road, Suite G7, Don Mills, ON, Canada M3B 1Z3. TPUG Magazine welcomes freelance contributions on all aspects of Commodore computing. Contributions should be sent on disk, though accompanying hardcopy is welcome. Be sure to include return postage if you wish materials returned. Please indicate on the disk label which Commodore disk format and word processing program you have used. Payment for articles published is $30.00 per page if the author retains the copyright, and $40.00 per page if the copyright is assigned to TPUG Magazine. Payment is made on publication. All contributions are subject to editing for length and readability. Address editorial contributions and related correspondence to: The Editors, TPUG Magazine, 101 Duncan Mill Road, Suite G7, Don Mills, ON, Canada M3B 1Z3.

Circulation
Subscription: 12,000 Newsstand: 10,000
ISSN #0825-0367

VIC 20, Commodore 64 and SuperPET are trademarks of Commodore Electronics Ltd. PET is a registered trademark of Commodore Business Machines Inc. CBM is a registered trademark of Commodore Electronics Ltd.

Subscriptions to TPUG Magazine may be obtained by joining the Toronto PET Users' Group (TPUG) Inc.

Regular member (attends meetings)	$35.00 Cdn.
Student (full-time, attends meetings)	$25.00 Cdn.
Associate (Canada)	$25.00 Cdn.
Associate (U.S.)	$25.00 U.S.
	$30.00 Cdn.
Associate (Overseas — sea mail)	$35.00 U.S.
Associate (Overseas — air mail)	$45.00 U.S.

For further membership information, please contact:

TPUG Inc. Membership Information

101 Duncan Mill Road	1552 Hertel Avenue
Suite G7	Suite 144
Don Mills, ON	Buffalo, NY
Canada M3B 1Z3	USA 14216

TPUG Telephone Numbers
Business Office (416)-445-4524
Magazine Office (416)-445-4524
Advertising Sales (416)-445-4524

DIRECTORY

86 TPUG Magazine #23

Inside Information

TBUGs

In last months issue, M. Garamszeghy's program **MFM Formater** contained an error due to the renumbering of lines during the typesetting. Line 30 should read

```
30 input"select a format";
   f:restore(f*10+90):
   read b6,bs,sd,f$
```

We apologize for any inconvenience and promise to watch out for those calculated **restore** statements from now on.

First offerings

In 1985 Commodore released it's long awaited successor to the very popular C-64, the C-128. More than just a 'fat 64', the C-128 offers a number of enhancements and options never before seen on the 'low end' Commodore computers. As in the past, TPUG intends to offer full user support in the form of monthly chapter meetings and public domain libraries. In the past, chapter libraries have started from scratch and have been built on the contributions of our members. However, with the implementation of the CP/M operating system on the C-128, a wide variety of existing public domain programs became available. Through the generosity of Lorne Gould and John Matheson of First Byte Software, a large library of these programs have been made available to TPUG members. Formerly a Toronto area distributor of CP/M public domain programs, First Byte currently manufactures and distributes the **$MAN Money Manager** for CP/M and MS-DOS operating systems. Lorne and his staff have generously allowed us to begin translating into Commodore disk format the megabytes of software housed at their Richmond St. offices and have given invaluable advice on configuring these programs for the C-128. These contributions will help establish the TPUG CP/M library as a major source of C-128 CP/M programs. As C-128 co-ordinator and CP/M librarian I would like to take this opportunity to thank Lorne and all his staff for their help, patience and professional considerations.

This month

This month we focus on the increasing number of alternative languages for Commodore computers. After years of making do with the relatively sparse BASIC built into Commodore computers, better BASICs have started making their appearance, as noted in articles on Amiga BASIC by Chris Johnson and 128-based BASICs by Adam Herst. A better BASIC for the C-64, PETs and VIC 20 is available in the form of Northcastle BASIC, a public domain BASIC. Yet another popular public domain language for the C-64 is COMAL, who's advantages are duly noted in an article by Len Lindsay.

Newer, less familiar languages such as Pascal, Promal and C are explored in articles by Dave Powell, Nick Sullivan and Ajay Jindal respectively. Overviews of programming in these various languages are provided by Avy Moise and Eric Giguere.

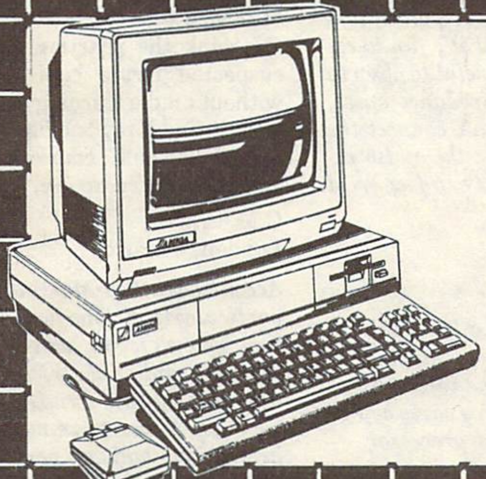
Next month we look at orphaned computers, those unfortunate machines no longer considered the cream of the crop. Support and information for these machines has become increasingly hard to find even though they may have years of productive use left in them. Our coverage will include such gone but not forgotten machines as the VIC 20, C-16, Plus/4, B-128 and the PET series of computers.

Dept. of Labour report

Last issue we revealed to an astounded world that two members of the magazine's editorial staff were expectant fathers. This is now a half-truth. After an extended development effort of nearly a year, Alexander Patrick Grantham was released to the public domain at 2:38 in the morning of April 14. The new unit comes equipped with a real-time clock that generates non-maskable interrupts at three-hour intervals, completely interactive i/o facilities, one-channel sound generation capability (upgradable to include speech synthesis real soon now), quite a few internal drives and an intelligent, fully multitasking operating system. Oh yes, and $0014 fingers and toes too. Congratulations to Tim and Cate. □

ELECTRONICS 2001

AN AUTHORIZED COMMODORE SYSTEM DEALER SINCE 1979



- New Products constantly arriving
- If a product is available We have it
- All hardware in stock at all times
- Full service

THIS MONTH'S SPECIALS

PRICES IN EFFECT UNTIL MAY 31,1986

THE MANAGER C-64 **$14.95**

OUBLIETTE **$14.95**

SIDEWAYS FOR C-64 **$39.95**

IMAGE MAKER MPS 802 HIGH RES. PRINTING **$10.00**

Compuserve Starter Kit **$39.95**

$21.95
The complete guide to the 1541 operating system.

SPRINT PRINT FULL SPOOLING PRINTER BUFFER
SALE PRICE **$19.95**

$21.95 Boss Joystick

PaperClip (the word processor) **$59.95** with SpellPack $79.95

Kraft Joystick **$15.95**

$36.95

ELECTRONICS 2001 LTD.

Name
Address

☐ Check or money order enclosed ☐ Visa ☐ MasterCard ☐ Amex

Acct # Exp. Date Signature

5529 Yonge Street, Willowdale, Ontario M2N 5S3
Tel: (416) 229-2700

Note: All prices in Canadian Funds. Phone and mail orders welcome.
Ontario residents add 7% sales tax.
Add 5% for shipping (minimum charge $4.00)
10% discount for TPUG members on regular priced software, accessories and magazines.

LINE NOISE

TPUG Magazine invites you to express your views on Commodore computing by writing to:

Line Noise
TPUG Magazine
101 Duncan Mill Road, Suite G7
Don Mills, Ontario M3B 1Z3
Canada

SID quirks

I have read your article in the November '85 issue of *TPUG Magazine* dealing with the SID chip. In your column 'SID'S Curious Quirks...', I was interested in the fact that you can route an external audio signal through the filters, controlling the loudness with the volume nybble. I would consider myself an amateur programmer in that I have mastered BASIC programming and can readily adapt to most of the special features offered by the Commodore 64, such as music and sound synthesis. I had already decided that access to the SID was not worth the effort in BASIC, so I have invested quite a bit of money in commercial software that specializes in music composition and SID access through the use of other, more convenient, languages. The idea of routing an external signal through the SID seems worth exploring, and I am requesting some further literature in how to do this.

Dean Niquette
Kaukauna, WI 54130

As far as I know, there is *no literature. The information in the article came from personal experimentation and careful reading of the SID chip specifications contained in the* Programmer's Reference Guide. *There is nothing I can add to that article except to tell you that I used a Casio VL-Tone mini-keyboard as the audio input. Connecting it as described in the article, I was able to mix its signal into the output of the SID chip. It's just like adding another voice to the SID chip: its loudness is controlled by the master volume nybble at $D418 (decimal 54296); and if you set bit 3 of register $D417, you can control its tone with the filters. Check to see if your music software can use this feature.*

The external input capability was originally intended as a way to chain the output from several SID chips contained in a piano-style keyboard. It's of limited utility, in my opinion, but it's fun to experiment with. Just be careful to observe the input voltage and impedance specs, and whatever you do, don't *connect the audio output of the SID to the audio input, or you'll probably have one deep-fried chip in your C-64!*

Tim Grantham

WP64 vile zap

The following is a copy of a letter sent to Pro-Line Software in Mississauga, Ontario. We publish it for the information of other users who have very early copies of Pro-Line's ***WP64*** *word processor.*

Boy am I mad at you guys!

I use my Commodore 64 in my consulting practice. The biggest use I put the machine to is word processing. At present, I am using version V.841115 of **WordPro 64** (or, as the program is sometimes known, **WP64**, or **The Wordprocessor**). The manual is the 2nd printing (September 1984 version). I also use **Spellpro64** (version number unknown, purchased December, 1984). I own a Juki 6100 printer, a Star SG-10 printer, a Cardco Card?+G printer interface, a BusCard II IEEE and printer interface connected with an MSD SD-2 drive as device number 8, and a 1541 as device number 9.

Today I attempted to load **WP64** from drive 1 of device 8, with my text disk in drive 0 of device 8. **WP64** did not load. Then it reset the machine and reformatted my data disk in drive 0 with the label 'PIRATE COPY'.

Of course, I keep backups of my data disks. Unfortunately, my backup was two days old, so I am out two days of work (or $600.00) getting back to where I was before I made my simple mistake. I reviewed the manual, and it does say the following:

> Place the WORDPROCESSOR program disk into drive 0 of device 8 and close the drive door. NOTE: You must use that specific disk drive. The WORDPROCESSOR can not be booted from any drive other than drive 0, device 8.

However, there is no warning that if you try to boot the program from any other drive it will reformat the disk in drive 0 of device 8.

I think the practice of reformatting suspected pirate copies of your disk without ample warning is poor software design, and if my backup had been much older, I would consider legal action against your company.

C.M. Smythe
London, Ontario

According to Stew Martin of Pro-Line, the particularly vicious form of copy protection you describe ***was*** *present in the earliest copies of* ***WP64****, a fact that somehow escaped Pro-Line's notice. Apparently, the programmer who added the protection had a perverse sense of humour. Pro-Line corrected the fault as soon as it was brought to their notice, and will replace the earlier ones on request.*

First impressions

Commodore's new machines are very nice, though one does encounter the odd quirk. For instance, the **CAPS LOCK** key works on all the alphabetical keys except the **Q** key. This is not too annoying, however.

The MSD SD-2 dual disk drive seems to work fine with the C-128 in any mode. The Batteries Included 80-column card does *not* work in C-128 mode, but at least it doesn't force the C-128 into C-64 mode upon initialization.

Commodore's 'new' MPS-1000 printer for the C-128 is actually the Epson Spectrum LX8064 printer with Commodore labels on it. This is a compact, fairly fast (about 100 cps) printer that apparently will do both hi-res bit-mapped graphics and good correspondence quality print (better than on the 1526/MPS802). The Spectrum LX-8064 has a 16 cps that comes very close to letter quality. It is slow but the quality is incredible!

The 1571 seems to be a pretty amazing and *extremely* versatile disk drive. According to Jim Butterfield, there will not be a dual drive version of the 1571. That's *extremely* depressing, for dual drives are far more useful than single drives, and can easily justify their added cost.

Sean Rooney
Port Credit, Ontario □

ATTENTION

ALL COMMODORE 64, VIC 20, COMM. 16 AND COMMODORE 128 OWNERS

A complete self-tutoring BASIC programming course is now available. This course starts with turning your computer on, to programming just about anything you want! This course is currently used in both High School and Adult Evening Education classes and has also formed the basis of teacher literacy programs. Written by a teacher, who after having taught the course several times, has put together one of the finest programming courses available today. This complete 13 lesson course of over 220 pages is now available for the COMMODORE 64, VIC 20, COMMODORE 16 and the COMMODORE 128and takes youstep by step thru a discovery approach to programming and you can do it all in your leisure time! The lessons are filled with examples and easy to understand explanations as well as many programs for you to make up. At the end of each lesson is a test of the information presented. Furthermore, ALL answers are supplied to all the questions and programs, including the answers to the tests. Follow this course step by step, lesson by lesson, and turn yourself into a real programmer! You won't be disappointed!

We will send this COMPLETE course to you at once for just $19.95 plus $3.00 for shipping and handling (U.S. residents, please pay in U.S. funds). If you are not COMPLETELY satisfied, then simply return the course within 10 days of receipt for a FULL refund.

FOLLOW UP COURSE

Now available! a 200 page course exclusively on sequential and relative files using a unique approach for those with very limited file programming experience - set up your own personal and business records! — disk drive a must — same author — same guarantee — same cost— this course for all computers except Vic 20.

Fill in the coupon or send a facsimile.

TP

NAME:__________

ADDRESS:__________

CITY:__________

PROV./STATE:__________

POSTAL/ZIP CODE:__________

I desire the BASIC programming course for:
Commodore 64 ☐ Vic 20 ☐
Comm. 128 ☐ Comm. 16 ☐
I desire the FOLLOW-UP ☐ course on relative and sequential files for all above computers but Vic 20).

Any complete course:	**$19.95**
Postage and Handling:	**$3.00**
Total:	**$22.95**

Send Cheque or Money Order to:
Brantford Educational Services
6 Pioneer Place,
Brantford, Ontario,
Canada N3R 7G7

---

shoebox BUSINESS SOFTWARE NEW

DIRECT FROM THE AUTHORS WITH A 30 DAY SATISFACTION GUARANTEE PLUS CONTINUING SOFTWARE SUPPORT

PROVINCIAL PAYROLL

FAST, USER FRIENDLY WITH HELP KEYS, SOUND CUES AND AN EXTENSIVE MANUAL • USE WITH C-64 OR C-128 AND COMPATIBLE DISK DRIVE AND PRINTER
• PAY JOURNALS • PAY RECORDS • T4 SLIPS
• CHEQUE WRITER • 99 EMPLOYEES

NOW ONLY 79.95

Qty. ___ x $79.95	$ ____
Shipping & Handling	$ 5.00
7% Sales Tax (Ont. residents only)	$ ____
TOTAL	**$** ____

DEALER INQUIRIES WELCOME

Prepaid Orders only

Name __________
Address __________
City _____ Prov. _____ Code _____

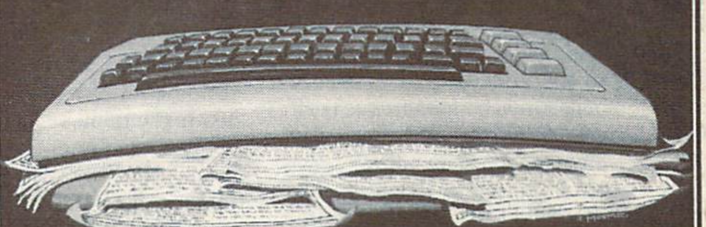
VISA MasterCard

JOHN DUNLOP & ASSOCIATES LTD.
R.R. #5 ORANGEVILLE, ONTARIO L9W 2Z2 (519) 941-9572

☐ CHARGE MY VISA OR MASTERCARD NUMBER | | | | | | | | | | | | | | | |

EXPIRATION DATE ___ | ___

Expand Past Maximum Capacity!



The Transactor

The Tech/News Journal For Commodore Computers

At better book stores everywhere! Or 6 issues delivered to your door for just $15.00 (Overseas $21 U.S. Air Mail $40 U.S.)
The Transactor. 500 Steeles Ave. Milton, Ontario. L9T 3P7.
416 878-8438
Also check out The Transactor Disk and The Complete Commodore Inner Space Anthology - to us, expansion knows no limits!

A Tour of Babel

by Avygdor Moise

I still remember standing in line with a deck of IBM cards, waiting to submit a 'job' to the main frame computer. My jobs would normally take less than one second to execute, and at the time I did not mind waiting for up to 30 minutes for my turn to submit the cards to the *card reader*, then wait up to an hour before the *print-out* was available. Normally, my programs would not execute correctly (if at all) on the first try, so I had to spend days debugging the programs before I could benefit from the results that were generated during a few milliseconds of computer time.

In that dark age of computing, I was never concerned about improving the program's efficiency by choosing the right programming language, or by using various programming techniques to speed up the execution of my programs. Efficiency and style was not an issue, since I did not have a wide choice of programming languages and the elapsed time between jobs far exceeded the CPU time of any given run. During the busy season, I had to wait up to a week before I could collect the printed results of a ten minute job.

Those of you who think that I must be ancient, since I was brought up on punch cards and paper tape, should note that the events described in the preceding paragraphs took place less than ten years ago, during the period of 1976-1979.

In the late 1970's, programmable calculators and micro computers became available to the general public and the concept of a personal computer was born. With the reduction in the computer's size, there was also a decrease in memory size, the availability of peripheral devices, and software. In addition, the computer manufacturers and software developers had to adjust to the fact that the average computer user was no longer a computer expert or a system programmer, but an inexperienced person with little or no previous exposure to computers.

The popularity of the micro computer can be partially attributed to Commodore. Commodore was among the first to produce a low-cost, user-friendly micro computer — the PET 2001. One of the most important features included with the early home computers was the development of an interactive operating environment/language called BASIC.

Without BASIC, I believe that personal computers could not have gained any ground in our homes and in schools. This is because BASIC provided a simple way to command and control the computer and its resources.

As micro computers become more powerful and approach the capabilities of some of the so-called main frames of the 1970's, the number of programming languages available for the micro computer is also increasing. This presents a problem for the modern computer programmer, who must choose from a wide selection of available languages. The right choice can make program development and maintenance a pleasurable experience. The wrong choice, however, may turn the simplest project into a nightmare (and I've had a few).

What is a programming language?

A programming language is a set of syntactical rules that are applied towards the construction of instructions to command the computer to perform a given task. A program is a collection of such instructions (statements), put together to perform a given application.

```
10 let x = 3
20 print x
30 end
```

This program consists of three statements (line 10 through 30) which prints the number 3 on the screen. Each of the statements was constructed according to the syntax rules that are required by the BASIC language interpreter.

When you type the command **run**, this program will execute, i.e. it will be translated to something like:

MOVE.L #3,X(A6) ;store the value 3 into variable X

MOVE.L X(A6),D0 ;convert the integer in X to an ASCII string

TRAP 0 DC.W M$ITOS TRAP 0 ;display the ASCII string

DC.W I$PUTS TRAP 0 ;terminate the program

DC.W F$EXIT

Are you confused? You should be!

The program segment listed above represents the code that may be produced by a BASIC translator (compiler) to reduce the syntax of BASIC to the computer's native code. The translation of a high level syntax (like BASIC) to machine executable code is done invisibly by the computer, right after you type the **run** command. Hence the term 'BASIC interpreter'.

Sooner or later you will find out that programs that are executed by an interpreter run relatively slowly. This is mainly due to the fact that the interpreter repeatedly converts the English commands that make up your program into machine code. The overhead that is generated by the dynamic translation of the program is large enough to slow down the execution speed of any program.

To speed things up, one may choose to write the programs in machine language. However, if this article was written in machine language, it would look like:

```
01001001   00100100   10100101
00101010   01001111   11101001
00100010...
```

Not a pretty sight. Considering that interpreters are slow and binary code is too hard to master, we use compilers as an alternative. High level languages, therefore, provide a tool for writing program code in a readable, easy to understand form, which is then translated by a compiler to pure binary code executable by the microprocessor. If the compiler is a good one, and the computer language is properly chosen, the object code produced by the translator will be, at times, as efficient as the corresponding code that had been hand-written at assembly level .

Hence, in acquiring a programming language for your micro computer you should

• Choose the right language (this may be application dependent);

• Choose the compiler that produces the

Continued overleaf. . .

Statement description	C	Pascal	Fortran
Data types			
Simple variables			
integer	int a;	var a:integer;	integer a
character	char b;	var b:char;	character*1 b
float	float c;	var c:real;	real*4 c
Pointers			
integer	int *a;	var a:^integer;	N.A.
character	char *b;	var b:^char;	N.A.
float	float *c;	var b:^real;	N.A.
integer function	int (*f)();	N.A.	external f
Arrays			
integer	int a[10][20]	var a:array[0..9,0..29] of integer;	integer a(10,20)
character	char b[100]	var b:array[0..99] of char;	character*1 b(100)
Structures			
record containing i	typedef struct x_	x = record	N.A
of type integer and	{	i:integer;	
c of type character	int i;	c:char	
	char c;	end	
	} x;		
User extensions			
structures	x y[2];	var y:array[0..1] of x;	N.A.
macros	#define INTEGER int	N.A.	N.A.
	#define f(x) g(x)	N.A.	N.A.
conditionals	#ifdef A	N.A.	N.A.
	X		
	#endif		
	#ifndef A	N.A.	N.A.
	Y		
	#endif		
include files	#include <stdio.h>	N.A.	include more.for
Operators (arithmetic)			
assignment	a=3;	a:=3;	a=3
addition	b+=3;	b:=b+3;	b=b+3
subtraction	c=5-b;	c:=5-b;	c=5-b
Operators (logical)			
test for equality	a==3	a=3	a.eq.3
test for not equal	b!=5	b<>5	b.ne.5
test for greater	c>b	c>b	c.gt.b
Operators (boolean)			
boolean and	a&&d	a and d	a.and.d
boolean or	b¦¦c	b or c	b.or.c
boolean not	!c	not c	.not.c
Operators (bitwise)			
bitwise and	a&3	N.A.	syntax is the
			same as boolean
			but data type
			must be integer
bitwise or	b¦5	N.A.	
bitwise not	~c	N.A.	
Conditionals			
if statement	if(T)	if(T) then	if(T) then
	S1;	S1	S1
	else	else	else
	S2;	S2	S2
			endif
	(expr?a:b)	N.A.	N.A.
Grouping			
statements	{S1;S2;S3...Sn}	begin	N.A.
		S1;S2;S3...Sn	
		end	
loops	while(T) S;	while(T) then S	do while(T)
			S
			enddo
	for(i=1;i<n,i++) S;	for i:=1 to N do	do i=1,10
		S	S
			enddo

most efficient object code for your computer.

In this article, I will concentrate on choosing the right dialect, since the choice of a compiler for a specific language and computer strongly depend on software availability.

Choosing the right language

Simply put, a good language should provide statement syntax rules that can be efficiently used in program construction and readily converted to machine code. Optimally, a high level language should support the following features and capabilities:

- Sequential execution of instructions and statements;
- Non-sequential execution of instructions and statements (branching);
- Conditional execution of sequential instructions and statements;
- Concurrent execution of instructions (multitasking);
- Grouping of statements (blocks and procedures);
- Flexible data typing (attributes that a variable may have);
- Memory manipulation of data and program object code;
- Input and output capabilities;
- Clarity, self consistency and reliability;
- Portability;
- Extensibility (user defined syntax).

Let's expand on some of the points in the above list. I will to use examples where possible to illustrate a certain feature.

In the examples that follow, I chose to compare three computer languages (C, Pascal and Fortran) that are known to run on most Commodore computers.

Note: The table does not cover all the features and capabilities of the languages. Only a few features were selected for the purpose of illustration. For a better understanding of the languages, please consult a user reference manual.

On the surface it seems that the three languages hardly differ from each other. This is true if your application requires only a subset of the language to solve the problem at hand. This is one of the reasons for the acceptance of BASIC as the major programming language by most users.

As seen in the table above, all three languages fully support sequential, branching, and conditional statements, and grouping. Only C, however, supports conditional execution of sub-expressions within one statement. For example,

```
a = (b>c?b:c) + 5;
```

will assign 5 plus the larger of b or c to a.

Also, C provides a larger selection of operators. This may be viewed as a useful feature that allows the programmer to use the optimal combination of operators for the application at hand. On the other hand, this may prove quite confusing to the beginner and may lead to the generation of inefficient code if the wrong construction is chosen.

For example, to increment a variable by one unit, any one of the following statements may be used:

```
a++;
++a;
a+=1;
a=a+1; ...
```

To reference the first array element we may write

```
a = *b;
a = b[0]; ...
```

Which one is the best?

If the languages are similar in most aspects, where do they differ? The main differences will be found in the way they treat the computer's memory. The type of data that can be stored and accessed by a program strongly depends on the data types that are supported by the compiler and the addressing modes that are allowed. It is obvious that Fortran cannot use or manipulate pointers to data structures, whereas C and Pascal can. On the other hand, Fortran can pass pointers to data and functions, whereas Pascal can pass pointers to data only. Therefore, if your application does not rely on pointer manipulation, any language will do.

In conclusion, it is not easy to choose the best language for your computer. This is mainly due to the fact that the best language for one application may be the worst for another. As a result, you may have to master a few languages to make your computer truly efficient. However, this also means that you may choose any language initially, for the purpose of learning, since once you have mastered one language, it is easy to master another. When deciding on the purchase of a computer language, your best bet will be to get the one that is commonly used on your type of computer, since this implies that it is likely to be relatively free of bugs. □

Missing Letter Puzzle

1) Insert your COMAL disk in drive*.
2) Type LOAD "C64 COMAL*",8
3) Type RUN (starts COMAL)
4) Type AUTO (turn on auto line#'s)
5) Enter the program lines shown below (COMAL indents lines for you)
6) Hit RETURN key twice when done
7) Type RUN
   Pr-gr-mm-r's P-r-d-s- P-ck-g-
   Programmer's Par?

```
0010 dim text$ of 39, disk$ of 2
0020 open file 2,"missing.dat",read
0030 disk$:=status$; count:=0
0040 if disk$="00" then
0050    count'text
0060 else
0070    close // no data file found
0080    create'text
0090 endif
0100 play'game
0110 //
0120 proc count'text
0130    while not eof(2) do
0140       read file 2: text$
0150       count:+1
0160    endwhile
0170    close
0180 endproc count'text
0190 //
0200 proc create'text
0210    open file 2,"missing.dat",write
0220    print "input text (or blank):"
0230    repeat
0240       input text$
0250       if text$>"" then
0260          write file 2: text$
0270          count:+1
0280       endif
0290    until text$=""
0300    close
0310 endproc create'text
0320 //
0330 proc play'game
0340    open file 2,"missing.dat",read
0350    for x:=1 to rnd(1,count) do
0360       read file 2: text$
0370    endfor x
0380    close
0390    for letter:=1 to len(text$) do
0400       if text$(letter) in "aeiou" then
0410          print "-",
0420       else
0430          print text$(letter),
0440       endif
0450    endfor letter
0460    print
0470    for letter:=1 to len(text$) do
0480       while key$<>text$(letter) do
0490          print "?"+chr$(157), //left
0500       endwhile
0510       print text$(letter),
0520    endfor letter
0530 endproc play'game
```

* If you don't have COMAL yet, order a Programmer's Paradise Package-$19.95. It includes the complete COMAL system plus over 400 pages of information. Add $5 more to get our 20 interactive lesson Tutorial Disk. Add $2 shipping. Visa/MC or US funds check accepted. Send to:

COMAL Users Group USA
6041 Monona Drive, Room 109
Madison, WI 53716
phone 608-222-4432

Structured Programming

by Eric Giguere

Structured programming has become the latest rage in programming methodology, even though it's been around for a long time. It's really the way a person should be taught to program in the first place, but the preponderance of non-structured languages in the marketplace (the most obvious example being BASIC) has made a structured approach awkward and limiting. I know that if you had told me a couple of years ago that I'd become a believer in structured programming, I probably would have laughed. Well, folks, I've learned.

What is it?

Ask three different people what structured programming is, and you'll probably get three different answers. Still, there are some basics that everyone can agree upon, and thus my definition of structured programming goes as follows:

> *Structured programming is a programming methodology in which programs are written in a hierarchical form with step-wise refinement and an evident program structure.*

For those who consider this to be a bit wordy, what it all boils down to is this: structured programming requires good planning and makes plentiful use of program modules. *Step-wise refinement* refers to the fact that, as you program, each part of your program is refined from the general to the specific. That is, you go from a general structure to a very detailed one, working on each step separately until the desired effect has been achieved and the bugs worked out.

In general, the concept of structured programming implies three things:

- The problem has been properly defined.
- No actual programming is done until a plan of action has been thought out.
- The program is written so that it flows logically from one module to the next.

This last comment refers to the fact that a good structured program will have no need for **goto**s or the like (exceptions arise when you try to simulate certain programming structures in a non-structured language). While this may seem to go against the grain of programming in BASIC, it can be done quite efficiently.

The Power of Planning

Perhaps the most important advantage of structured programming is that it requires a lot of planning. Two years ago I probably would have attacked a programming project by sitting down at the computer and programming off the top of my head. This method works — and is still useful for me when I need to write a small, quick and dirty program for temporary use — but has its disadvantages. You may end up spending more time in finding and correcting errors than in entering the original source code; and goodness knows how much fun you would have trying to modify the program.

In contrast, structured programming requires you to define the problem, create a general algorithm for solving it, and refine each step in the algorithm. You can, of course, go into great detail while planning a program, reducing the job at the end to merely typing in the code and testing it, or you may leave it in a more generalized form and create the code at the keyboard, according to the plan you have conceived. There are no hard and fast rules for structured programming — individual preferences and style play a large part in determining what you do.

While this planning may seem a bit tedious to some, there are definite advantages that, in my opinion, outweigh the disadvantages. First and foremost is the fact that you have *defined the problem.* While this may seem obvious, many programmers start with only a vague idea of what they would like the program to do. Knowing exactly what it is you want to do allows you to be more precise in your coding and to discover hidden pitfalls in the task you are attempting.

The second advantage lies in the fact that a structured and well-planned program is much, much easier to modify and extend. The program's structure should be obvious and the variables well-documented so that you know precisely what a section does and what you must do to properly change it. Of course, certain languages make this much easier to do, as we shall see below. Third, program bugs are easier to locate and correct. Finally, the program's structure makes it easier to read and understand by others, especially if the code is well-commented. For these reasons, the time spent in properly planning a program is time well spent.

Languages

While structured programming practices should be implementable in almost any language, some languages are more conducive to it than others. Pascal, for example, is probably the most-cited example of a structured programming language. Programming in Pascal requires a structured approach, with much emphasis on well-conceived program modules. C is another such language (my current favourite). C is more flexible than Pascal, but also requires a structured approach to programming. Other languages, such as FORTRAN or BASIC, are decidely unfriendly to structured programmers, and properly implementing your plans can be a real pain. Since learning C, I've often wished that BASIC had **while** loops (structures that repeat themselves until a certain condition is met) or **case** statements (for making multiple choices), and sometimes I mistakenly program using them. Even an **if...then...else** statement would be nice! It's hard to go back to the old ways once you've started seriously using structured programming methods.

If you would like try your hand at structured programming, you can do so by purchasing one of the several C or PASCAL compilers available for Commodore computers. A good introductory book on the language is also recommended. If you prefer not to invest much money in a structured language, another alternative presents itself: COMAL. COMAL is a hybrid language that has elements of both BASIC and Pascal, making it easy for BASIC programmers to learn while giving them powerful structures to work with. Its best feature is its price, nothing! You can get COMAL 0.14 (the public domain, disk-based version) from TPUG or from The COMAL Users' Group and other sources. Tutorial and reference books are also available, as well as an extended cartridge version of COMAL (this one *isn't* free). This is perhaps the best and easiest way I know of to gain exposure to structured programming. If you're interested, I'd suggest you try it. □

Four Pascal Compilers for the C-64

by Dave Powell

Oxford Pascal

Oxford Computer Systems, Oxford, England. Distributed in North America by Limbic Systems, Inc.

I reviewed this package in the January 1985 issue of *TPUG Magazine*, so just the highlights here. It turns out that Oxford has a pretty good product. The major advantage, specially for new Pascal users, is that it has an *in memory* mode (the editor, compiler, source and object are all in memory) for quick compiles of small programs. Although some features are missing in this mode, it is great for experimenting — to get the syntax right, for instance. Without disk access requirements, compilation is very fast.

The major flaw in the product is the fact that the compiled code is slow. I believe that p-code, rather than machine code, is produced. This also means that even a small program is saved as over 50 blocks if it's wanted in a form that will run from BASIC.

The program development environment is good because of the no-disk compile. All the errors can be found in one compile. Machine code can be incorporated, although there is no built-in assembler. Documentation of internals is sketchy.

This is a full implementation of the language, with a good set of extensions. Strings have only basic support. Much of the built-in C-64 operating system is used, so often a familiar poke will still work.

Other program code can be included: there is a source include, program chaining, and linking of separately compiled programs. All in all, this package is still a good choice, unless you're after machine language speed.

Zoom Pascal 64

King Microware, Quebec.

This compiler has the advantage of being inexpensive: full price was around 50 dollars (Cdn.) and I found a clearance copy at 10 dollars! Its big fault is that it is a disk-based compiler, and stops at the first compile error. This makes program development *very* tedious. Because of this, the amount of experimenting that I did was limited, and much of the following is based on the documentation, rather than experience.

An editor is supplied, which adds some functions over the built-in C-64 screen editor. Beware of losing files using this editor. The original file is opened for update, and the result of your edit replaces the original upon exit. Tough, if you've deleted all lines, as I did. It is apparently possible to define editor macros, but the manual stops short of a full explanation. The compile is a two-step compilation to p-code, which is then translated to machine language.

Implementation is a subset of standard Pascal, with extensions. String handling functions are more extensive than any of the other compilers I've tried.

I'm not sure how well C-64 features can be accessed. A **mem** array is equivalent to **peek** and **poke**, so sound and colour should be available. Object code is ML, and a stand-alone program stores in 30-plus blocks. File access appears to be normal, and a sample program on the disk gives an example of relative file programming. A **call** procedure is provided, apparently for using external ML routines, but is not explained. No other form of program include is available. This is a cheap, but potentially frustrating, introduction to Pascal.

PASCAL 64

Abacus Software, Inc., Grand Rapids, Michigan.

Judging solely by the recent price reduction, I would say that this Pascal is being replaced by **Super Pascal** in the Abacus line. In a recent catalogue, the price is $12.95 (US) 'while quantities last'! This compiler has one of the better 64-specific sets of extensions, creates ML object code, and has good string functions. On the minus side, the development environment is very inhibiting. The compiler stops at the first error, issues a cryptic **compile error in xxxx** message to the screen, and then jumps to the cold start routine a few seconds later! This means that if one is not watching intently, not only is there no result to run, but the compiler has to be reloaded and rerun just to establish the line in error.

Because of this, and the fact that **Super Pascal** was introduced, I didn't dig too much deeper into this product. Incidentally, the 'cold start' route is undoubtedly an anti-piracy measure. Since **Pascal 64** uses the built-in screen editor, rather than its own, the designers had to expunge their code from memory before handing back control! Another example of how piracy hurts everyone.

I like the idea of using a familiar editor (it also means that **Power 64** and similar aids are usable), but it has its restrictions. The character strings **rem** and **data** must never appear anywhere in a program, even embedded in other text. Also, indenting presents problems: a semicolon must be used in the first column, before spaces are accepted.

The compile is followed by a loader routine and a save. The loader has to be started with **run 100** — not friendly. External source code can be included by running a linker program.

Pascal-64 has *nearly all* of the features of standard Pascal. It also has procedures to plot and unplot graphic points, to create sprites, and to create and run an interrupt-driven procedure so that two routines will run coincidentally.

The compilers for **Zoom Pascal** and for **Pascal 64** have a common heritage: one is licensed from the other. However, they are no longer identical. **Pascal 64** now sports more functions, while **Zoom** has more development features. However, I cannot recommend any compiler that needs to be reloaded to find each individual error, as both of these do.

Super Pascal

Abacus Software, Inc., Grand Rapids, Michigan.

This is a developer's compiler. The major feature is the integration of machine language code within a Pascal program. Individual functions or procedures can be coded in ML. I have two large complaints, though. First, the only way to compile is to copy the source to the product disk; the object has to be copied off it later. This is an invitation to disaster, as there is no backup disk. Secondly, in creating a development environment, even the DOS has been changed. That's fine for stand-alone programs, but try transporting data to and from a spreadsheet!

According to the manual, two 1541s can be used. The compile procedure asks which drive to use, but fails if the source is not on '0' (unit 8). Not even the copying utilities work properly with two drives. A letter to Abacus produced this response: *The program will only compile using one disk drive. The two drives are used only for copying the programs in the utility section.* When I tried copying, one file copied, and the next failed. Compiles with two drives fail in different places each time. I think they have a timing problem in their custom DOS. Not being able to use two drives is an inconvenience, but worse is having to compile from the product disk. I never like writing anything to a protected disk. After using this one for a while, there have been occasions when a message floats across my screen saying that I shouldn't use a copy. Apart from being libellous, this is quite worrying.

Otherwise, the development environment is good. The compile identifies *all* the errors, and it's fast. Smaller programs can be compiled to memory, which is even faster. However, the compiler still has to be loaded from disk. The supplied editor has a move command and auto-indenting: each new line begins under the start of the previous one. The source code of the editor is supplied – it's in **Super Pascal** – so anything you don't like you can fix. This is a great idea – far better than some example programs that are supplied. Debugging one's program is aided by a variety of options for listing extra information: memory locations and p-code instructions for instance. There's also a *post-mortem* dump, which shows the current contents of the variables.

This is standard Pascal plus extensions. The extras make the ML environment more livable, and add file functions. There is a built-in definition for string types, and an **else** added to the **case** statement. I'd like to see some string functions added.

There are no facilities built-in for colour or sound, though a sample ML procedure is given to change the screen colour. Although there is a **peek/poke** mechanism, a **mem** array, this seems to address pure RAM, so a procedure is necessary to get to the VIC chip, changing byte 1 to switch memory. Rather a roundabout way to change colours!

There are all kinds of ways to go beyond the limitation of one source creating one object: overlay segments, program chaining, calling a separately compiled Pascal program as a subroutine, external ML subroutines, included source code, and continued (chained) source code! That should keep most programmers content.

The result of your pains, a tested program, is transportable without the product disk, but must be on a 'Super Pascal DOS' disk. It's machine language, not p-code. The special DOS is a blessing and a pain. It's claimed to be three times faster than normal (and could well be), but this means that Pascal and BASIC can't be mixed.

The best part of this system is the ML integration. There is a complete assembler, with a fair set of pseudo-ops including conditional assembly. As I mentioned, a function or procedure may be coded in ML, right in the middle of the Pascal stuff. The utilities included in the system complement the assembler with routines to move hunks of memory to and from disk, and others to view or print various pieces.

This is an excellent development Pascal, flawed in only two respects. If the problem with supporting two drives were cleared up, I would forgive it the amount of customization it has wreaked upon the C-64 architecture. **Oxford Pascal** is still the friendlier to a Pascal neophyte, but once one is over that initial hump, **Super Pascal** will probably be the more satisfying. □

Switch

to the highest quality, lowest priced brand name diskettes available in Canada.

Q. Who has switched?
A. Professional and educational institutions, national accounting firms, major engineering firms, software houses, computer retailers and the Toronto Pet Users Group.

Limited time offer
SSDD - $11.00/box
DSDD - $14.00/box
(Minimum purchase - 5. Delivery extra.)

In Bulk - 3½ inch discs
SSDD $3.50 each / DSDD $4.00 each

P·G **Programmers Guild Products Ltd.,** 255 ManuLife Place, 10180 - 101 St., Edmonton, Alberta T5J 3S4 Phone: (403) 428-6229 VISA MasterCard

What to look for in a Compiler

by Dave Powell

If you don't know what you're looking for — how will you know when you've found it? This was the question that occurred to me late one night after yet another disappointment in my ongoing search for the Perfect Pascal. That's when I decided to write this article. Possibly, when it's finished, I'll know what I'm looking for, and it should help you out there, even if you didn't know you were looking!

Here's what I want: a good development environment; a reasonably full and accurate implementation of the *official* language; simple access to the standard C-64 facilities, such as colour, sprites, sound and full-screen interaction; fast running object code, which can be run on any C-64 (without the protected disk); the ability to read and write standard C-64 files from a Pascal Program; enough information to allow a developer to insert ML routines if necessary; and the ability to create large programs from prewritten segments — preferably without having to compile them again.

Now, is that much to ask? Apparently it is. Two of the compilers I've seen don't even pass the first hurdle. Let's go into some more detail.

Development environment

I want a product that will give me quick turnaround for problem solving. For a short program, it should be possible to make a change, recompile and run in one or two minutes. If there is more than one compile error in my code, I want the option of seeing them all. In practice, this means compiling in memory, or using two drives, otherwise one ends up pumping plastic in and out of the slot. I do not like the *solution* requiring the code to be copied to the product disk.

The editor is critical. An ideal solution combines the standard 64 built-in facilities with extras like renumber, auto number and, preferably, a copy and move function. (The latter is particularly useful with Pascal, in which — because the compiler insists that everything be defined before it is used — one frequently has to shuffle procedures around as development proceeds.) A helpful feature is automatic indentation to match the previous line.

If you will be adding many ML routines, then a built-in assembler will come in handy. Such testing facilities as a trace, variable dump, and a table of ML addresses of each procedure, are very handy, because without the interpreter to make instant changes, you'll want to get as much information out of each compilation as you can. (The faster the compile, the less this matters, but it's still very different from BASIC.)

Lastly, it's important to be able to get listings of everything required. Little things, like being unable to print square brackets, can be very frustrating.

Full implementation

This requirement is flexible: a lot depends on how much program portability you need. Naturally, if there is a particular language feature you need, the compiler should support it. Most implementations will have a workable subset.

In Pascal compilers, most programmers will look for extensions that will save them work. Notably, string definition and manipulation is missing from the *standard*. Many compilers implement some kind of string support and, depending on how good it is, you will be more or less frustrated when it comes to writing home applications, rather than textbook examples.

C-64 features

Many programs written in the home will want to use C-64 features. What use is a colour monitor or TV if it needs an ML program to change colours? Not everyone wants to program the SID chip — does the compiler have built-in functions to help? How is I/O to and from the screen handled? Does the compiler allow reading to and writing from specific screen locations? If the answers are no, is it still possible for the programmer to provide these facilities? Is there an equivalent of **peek** and **poke**, and will the usual operating system memory locations still be valid? I like to see enough extensions written into the language to handle the majority of 64-specific needs.

Independent object code

What I'm looking for here is the ability to compile a program and put the object code on a new disk that I can sell, give away or whatever — it is independent of the product disk. More than that, a ten-line program should not take up thirty blocks on the disk! Some compilers create a run-time package that takes for ever to reload. Naturally, I'd like the object code to be efficient ML, not semicompiled code.

M/L interface

From time to time, one may need the speed of pure M/L. The compiler should allow a JSR into a routine, and the documentation should be sufficient to determine where to put it. Some compilers allow assembler code to be intermixed with Pascal source code — that's the ultimate in convenience.

Suppose you have a routine that will be useful in several applications. If you merge the source into each application, then any improvements in the routine will either get left out of the later applications, or will be a pain to update. In a compiled language, it's sometimes possible to include the *source code* for the routine at *compile time*, in which case the new routine will get picked up when the application is recompiled. With a dynamic load technique, on the other hand, the *executable code* for the routine is brought in at *run time*. This results in every application getting the same version (although in practice there may still be different versions on different diskettes). Each of these techniques has its advantages and drawbacks. A good compiler will allow the developer to use them alone or in combination.

Now, if I was going to be scientific, I would evaluate each attribute that I was looking for, accord it a weight in proportion to its importance to me, then grade each compiler on each attribute. The sum of the products of grade and weighing is the score. Then I would buy the compiler with the highest score. Did you see the snag? The only way to grade a compiler is to use it for a couple of months!

In the Pascal reviews that accompany this article, I've attempted to cover some of these points. A couple of compilers had such an unfriendly development environment that I did not persevere in using them. In those cases, much of the review is based on the claims of the manual. This is not necessarily an accurate reflection on the product. Have I found the right one for me? No, not yet, but some have come close. □

Language arts

by Jim Butterfield

A language is more than just a method of communicating information: it also involves transforming the information in some way. ASCII is a code, not a language: it doesn't modify the data in any way. When we express a thought in English, French, or Chinese, we adapt that thought to the 'mindset' of the language involved. The idea often comes through slightly modified by the language: translators often have difficulty carrying a thought from one language to another.

The same is true of computer languages. Your choice of Fortran, BASIC, COBOL, LOGO, machine languge, Pascal, C, Lisp, Modula 2, COMAL, APL, or Snobol (to name just a few) will influence the way you approach a problem. And most languages have dialects: for example, there are many BASICs to choose from.

From a human standpoint, there are two general quality levels of person/processor communications: poor and lousy. A simple question such as 'Got the time?' might produce:

?syntax error

Poor communication both ways. Or you might get:

command 'got' not in vocabulary

Better, but not much.

file 'the time' not found

At least it's trying.

the time at noon today was 12:00

This isn't too useful.

explain 'got'

This is going to take a while.

We have to reshape our thinking to communicate with a computer. The manner in which you type commands is a language, and so are the responses you receive (oddly enough, these input and output languages are different).

When a program runs, the information we type in is often a kind of language. This is especially true if we have a large number of options. Thus, your word processor's commands consitute a language, and some WP's have better languages than others. Spreadsheet programs, too, have their own languages. You need to become proficient in the use of these languages in order to use the programs efficiently.

Interpreters and compilers

The 'formal' computer languages -- Fortran, BASIC, and so on — are generally grouped into two types: compiled and interpreted. For those unfamiliar with the terms: a compiled language (such as Fortran) must be completely written and then translated ('compiled') before it can start to run; whereas an interpreted language is translated as it runs. Compiled programs run faster; interpreted programs start quicker. If a compiled program needs a change, it must be recompiled — a lengthy process; an interpreted program can be fixed and restarted in a few moments.

Early computers were expensive and could do only one task at a time. All programs were compiled for fast run time. Fortran and COBOL dominated the high-level language world.

In the mid-1960s, computers were still expensive, but multitasking became practical. One slow job wouldn't hang up the whole computer. An array of 'computer ports' allowed many users to be connected. A single 'interpreter' program could run many users' jobs at virtually the same time. The 'time-sharing' computer concept started in the field of education and spread to computer service companies. The first two interpreted languages were JOSS (later adapted to FOCAL, and now almost forgotten) and BASIC.

The BASIC languages we see today are greatly changed from the original BASIC, but the style is still recognizable. Good old, sloppy old BASIC... it's so universal in small computers that it seems unlikely that it will ever be superseded. There's both joy and pain in 'spaghetti code', and BASIC lets you write any way you want.

With the advent of personal computers in the late 1970s, interpreted languages — BASIC especially — became the standard. Interpreters took up less space in these small computers (there was even a 2K tiny BASIC!), and they were more friendly to beginners who were just discovering computers. The interpreter could be placed in ROM — read only memory, another technological innovation — so that the language would be in place the moment the computer was turned on.

Now we're going through a period of reflection and consolidation. There are now compilers for languages such as BASIC, which were designed to be interpreted. These can be only partially successful. For example, the design of BASIC leads to clumsy handling of some variables — especially small integers — and a compiler must preserve this awkward logic or risk being 'incompatible'.

On another front, 'dual' languages have been designed, such as Pascal, which are capable of being either interpreted or compiled. Even so, in practice it seems that commercial packages seldom do both jobs well... they always categorize themselves into one camp or the other.

I can recall asking one of the COMAL principals why they had no compiler for their language. He emphasized that COMAL was designed as an interpreted language, and that specific language decisions had been taken with this in mind. He was correct... but I still pine for the best of both worlds.

Which, when and why

I think that most people will choose a language based upon the kind of programs they want to write. For payrolls and billing statements, a language like COBOL is wonderful. For advanced engineering, Fortran has major advantages; artifical intelligence — Lisp or Prolog; text editing — Snobol; mathematically-oriented material — APL. The list goes on, and different programmers will have their own favourites. Some languages seem to be in the eye of a storm: for example, C is an object of both love and hatred.

But most of us can't go out and learn a new language every time we have a new program to write. We must develop skills with a few selected ones. And in most cases, the most prevalent language will turn out to be... BASIC.

This leads to the next question: which BASIC? But that's a question for another time... □

The C Language

by Ajay Jindal

First, a little bit of history. C is a member of the 'Algol family' of algebraic languages. It has more similarities to PL/1, Pascal and Ada than to BASIC, Fortran or Lisp. C was first implemented on a DEC PDP-11 by Dennis Ritchie at Bell Laboratories in 1972. At about the same time, the Unix operating system was being developed. The two were, in a sense, merged. Unix was rewritten in C, and a C compiler was a part of the Unix system. C's ancestry includes Algol 60 (1960), Cambridge's CPL (1963), Martin Richards' BCPL (1967) and Ken Thompson's B (1970), which was also written at Bell Labs. C is not a dialect of these languages, but it is a descendant of them.

C is a compiled language. You write source code using an editor, translate the source code into machine code with the compiler, before executing the machine code. Because C syntax was designed for easy conversion of program statements into machine language, this process makes for fast, efficient programs. In a sense, then, C is very close to ML. Don't get the impression that you have to memorize a lot of op-codes and memory locations, though — C has variables, loops, functions, arrays, and most of the familiar high-level statement types you probably know already from other languages.

There are usually two ways of writing code in C: a high-level and a low-level style. If you are just starting out, you would probably use the high-level style, because it is easier to understand and similar to other languages. However, compiling this type of code is somewhat less efficient. Using the low-level coding style requires you to understand the language very well, as well as the compiling method and the concepts of ML. Such code has an almost one to one correspondence to ML and is very efficient. For example, to increment a variable **count** by one, one could either write **count = count + 1** or **+ + count**. Most machine languages have an increment operator in their instruction set, so for the second version the compiler simply generates an increment instruction. As programmers gains experience with C, they can write their code so that it lends itself more and more to the compilation process. This permits a gradual transition from a high-level to a low-level language without having to learn about things like registers, stacks, program counters and relative branches. If, on the other hand, you would like to use the high-level style to make your program more understandable, C lets you do just that.

C is easy to learn. One can learn most of the basic features of the language very quickly, and start programming right away. Once you feel comfortable with C, you can start delving into more advanced features. C is a very small and compact language, with relatively few keywords (about 29; some versions have more). It also possesses a particularly rich set of operators but, again, beginners can get by with only a familiar subset of these. Among the statement types offered by C are looping structures like **while**, **for** and **do**, declarations for integer, floating point and character data types, and decision making structures like **if**.

Prominently absent from this list, as you may have noticed, are statements to handle input and output. Strangely enough, if you are accustomed to other languages, C doesn't have any! However, a standard part of every C system is a library of functions to perform I/O and other tasks for which the language itself does not provide. These functions are customized for the hardware that the compiler is running on so that the compiler doesn't have to worry about it. This feature aids in the relatively simple porting of code to different C compilers running on various types of machines.

Because the language is small, it doesn't take much time to learn all the keywords, but to wield the power of them fully takes a lot of practice. This is especially true for BASIC programmers, because so many new concepts are introduced that don't have a BASIC equivalent. In some respects, you have to forget about BASIC and start over again.

C is a portable language. A typical C program has no direct references to hardware registers, so in theory a given C source program could be compiled and executed on any computer for which a C compiler has been written. Most C compilers are derived from a Portable C Compiler (PCC) written at Bell Labs. Because of this, they all compile identical source code into roughly identical machine code. Although hardware incompatibilities mean that C programs in general are something less than totally transportable, more than 95% of C source code can be compiled on other computers without modification. And a special facility called the preprocessor does provide a rational way to treat hardware-dependent items like screen memory. The preprocessor is responsible for scanning the source code before compilation, and executing certain special instructions — macro definitions, conditional compilation, inclusion of other source files and constant definitions. Hardware-dependent constants defined in preprocessor instructions can be altered easily when the time comes to port the program, without affecting the source code itself.

There is no 'official' standard for C, but most implementations follow the standard set in the book *The C Programming Language* by Brian Kernighan and Dennis Ritchie. In 1982, the American National Standards Institute (ANSI) formed a technical committee to formulate a standard for C, its library routines and execution environment. These standards aren't expected to deviate much from the 'unofficial' standard already in place.

C has some unique features. It has a strict syntax but no confining rules. You can do anything you want as long you don't break any syntax rules. For instance, if you declare an array of 50 elements and try to reference the 60th element in your program, C won't stand in your way — the compiler will not regard this as an error. Of course, the program will give unpredictable results. This puts an extra burden on the beginner, but a seasoned programmer may find a way to take advantage of C's relaxed attitude. Another distinctive feature is the use of pointers. ML programmers are already familiar with pointers, which provide a way of referencing a variable without naming it explicitly. Arrays belong to the high-level style of programming, while pointers are low-level. Actually, arrays in C are changed to pointers during compilation. Pointers are faster than arrays and are very important for creating data-structures such as linked lists and trees.

In C, every program module is a function. Even the mainline code belongs to a function called **main**. There is no distinction between a function and a subroutine or procedure — a subroutine is simply a function that doesn't return a value. Functions may receive arguments, but can only return a single value. The print 'command' in C is really a prewritten function in the standard C library.

In common with other structured languages, C allows both local and global variables. The programmer has great freedom in declaring where in a program a particular variable will be valid — the 'scope' of the variable. Sometimes, the scope will be only a few lines; most often it will be an individual function. Variables of larger scope may be shared by all the functions within a source file, or by all the functions in a program consisting of several files. The capability of declaring variables of narrow scope encourages the writing of modular programs, for you can guard securely against variable-name conflicts in a way that BASIC does not permit.

C has only three fundamental data types: integer, floating point and character. All other types must be derived from these three: a string, for example, is an array of characters. Records containing different fields of different data types can be constructed. The complexity of available data structures is virtually unlimited. In keeping with the ML tradition, bit manipulation operators are also available.

Another unusual feature of C is that statements can legitimately include other statements in places where most languages would insist on an expression. Actually, it would be more accurate to say that many types of C expressions are also complete statements. Here's an example:

```
printf( "\n %c %c",c =
    getchar( ), ++c);
```

This is a single statement (all statements end with a semicolon). But it also contains two expressions (**c = getchar()** and **+ + c**) that could be used as complete statements in themselves. In most languages, at least three statements would be required to take the place of this one.

C is a general-purpose language. It has been used to write operating systems, graphics programs, games, word processors, data base managers, business program and many other things. Its popularity comes from the fact that it makes both programs and programmers more efficient. Its execution speed can approach that of hand-coded machine language but, thanks to its portability, programmers don't have to learn a new instruction set each time a new microprocessor comes out. The speed difference between compiled C and human generated ML is negligible in all but the most demanding cases.

Implementations of C are already available for the Commodore 64, the Amiga and nearly every other computer. If you are seriously interested in writing fast, portable and easily maintained programs on your machine, you should be seriously interested in C. □

OS/9 Software NOW AVAILABLE!

Word Processor – Stylograph III

Mail merge, Spell checker. Menu driven, fast. Extensive features. Formats to screen as you work. . . . . US $149 Cdn $210

C Compiler

A full implementation of K & R standard C, with double precision variables.
. . . . US $138 Cdn $170

BASIC09

A fully structured, sophisticated semi-compiled (I code) BASIC. See Byte magazine April 1984. . US $110 Cdn $155

FORTRAN 77

Full FORTRAN compiler…coming soon.
. . . . US $120 Cdn $166

DynaStar, DynaForm, DynaSpell

A full screen (similar to microEDIT) editor, text formatter and spell checker.
. . . . US $90 Cdn $127

Other: Pascal, DynaCalc, Database etc., available on demand.

To Order: Send a cheque or money order (add $10 for shipping/handling and 7% PST where applicable) to:

TPUG,
101 Duncan Mill Rd., Suite G-7,
Don Mills, Ontario, Canada M3B 1Z3

Amiga BASIC

by Dick Barnes

*The **SuperPET Gazette**, which Dick Barnes edits, is the official newsletter of the International SuperPET Users Group, and is an excellent source of information on the SuperPET and much else besides.*

The buggy ABasiC for the Amiga has been replaced by a new Amiga BASIC from Microsoft. Commodore began distributing this new dialect with V1.1 of Amiga's operating software. I was considerably surprised to find that Microsoft has not only entered the 20th century with many structured features, but has also maintained compatibility with older versions of the language — no mean feat.

A flavour of the language

At last you may indent code. **While...wend** is built into the language. You may construct any structure wanted. Labels are allowed — and must be followed with a colon. Line numbers are optional (so is free speech in the USSR, and on the same terms, as we later see). The language converts all keywords to capitals, whether you like it or not. Variable names may be up to 40 characters long. **If...else if...else end if** is now allowed, in the form shown, with no limit on the number of **else ifs**. The **then** statement is required; **else** is optional.

There is a price for upward compatibility. Variable names aren't sensitive to case: **alpha**, **Alpha** and **ALPHA** are the same. You may use periods in variable names: **file name** fails; **file.name** is okay. There are no modern **mat** statements; all work on arrays must still be done in loops.

For compatibility, Microsoft hauled its archaic and clumsy method of handling REL files into Amiga. You have a bushel of separate intrinsic commands specific to random-access files, of which only two (record length and record number) are necessary for such work in a well-designed language. **Line input**, however, did see the light of day, although the simpler **linput** was lost along the way.

Amiga BASIC has two types of subroutines, one of which is a 'subprogram' with local variables and parameter-passing. Ordinary subroutines may be entered with a **gosub** to either a label or a line number. Subprograms may be **called**, as shown below. You have two options: in the first, you pass arguments by reference. This changes the values of **a**, **b**, and **c** in the main program by what is done to **x**, **y** and **z**.

```
CALL sendscreen(A,B,C) 'cre
    ate A,B,C values
SUB sendscreen(x,y,z) STATI
    C
if <condition> THEN EXIT 'm
   anipulate x,y,z
...statements
END SUB
```

In the second option, you may enclose the arguments in parentheses, as in:

```
CALL sendscreen((A),(B),(C)
  )
```

The values of **a**, **b** and **c** in the main program remain unaffected. In short, you have full control over whether a subprogram's variables are local or global. In addition, you may define any variable in a subprogram as a **shared** variable, which may be used in any other subroutine or in a main program, and which passes its value there. So far, so good.

Unfortunately, recursive subprograms aren't allowed. Do you see the **static** that follows the subprogram name above? It's required. Nobody in his right mind demands that subprograms be designated **static** if that's the only flavour you can define. And it is! **Static** (like other features of this BASIC) is derived from C. Static variables retain their value in a subprogram between calls. The other C option of automatic variables, which do not retain their value between calls, is not found in Amiga BASIC, but the presence of **static** indicates that Microsoft plans to use it in later versions. Either that or **auto** variables were taken out at the last minute. Anyway, because all subroutines are **static** and all variables retain their values, recursion becomes deadly; the manual warns users not to try. Too bad.

The manual comes with an errata sheet that asks you to remove several references to **reset** (which closes all open files). It seems they planned to have it and jerked it out at the last moment — along with the **auto** variables. We'll later see why.

Line numbers vs. labels

You are indeed free to use line numbers, which I prefer for reference when I list or edit. But there is utterly no way in Amiga BASIC to generate those line numbers automatically, or to renumber them thereafter. The language doesn't even check to see if line numbers conflict or are in proper order. Microsoft obviously doesn't want you to use line numbers.

How do you list a part of a program? Well, er... uh... you'd better stuff in named labels every 20 lines or so, or add line numbers on the same basis — or you must list your program from beginning to end to find the part you want. When we tried to list the subroutine below (without the label **test:**), Amiga BASIC refused. You must add either a line number or a label (as with **test:**) despite the fact that the subroutine is already named. And there is no search or search/replace in Amiga BASIC's editor, either.

```
SUB test STATIC
...
END SUB
```

Suppose you want to add a library routine from disk to a program on screen. You **merge** it at the end, then **copy** and **paste** into the program. Line numbers have no effect on a **merge**.

Directories

You're supposed to be able to get directories by saying, for example: **files "df1:basic.programs"**. Do you get the Amiga's typical two-column listings? Nah! You get a list in one column, and it scrolls right off the screen unless you are fast enough to hit **right-amiga s** to stop that listing. Golly, when will those folks in Belleview learn to pause listings at the end of each screen page? Sorry to say that half the files on our Workbench disk are never listed at all. It's a bug. Well, click over to DOS for directories...

Immediate mode

You may not cursor up and amend. We're used to working out difficult algorithms in immediate mode, by defining our constants and variables and testing variations in often-complex code. You won't do that in this BASIC: you must retype all the code each time you test, which is so time-consuming that you are far better off to make a program of the test. And — a **run** always clears the screen (you can't control that), which wipes out all previous results. Most of the utility of immediate mode is thus destroyed. We retreat to making pencil copies of data we've been handling on screen for years. Back to 1920.

You have two separate windows: one for program output, one for listings; you must interminably click your way back and forth between them. It seems this was done so that you can see your code execute in one window and trace the executing line in the other. We tried it for a couple of hours and conclude that the folks at Belleview are mad. The output window is always obscured by the list window; a little red border skips from one executing statement to another so swiftly that you can't follow it. If you try tracing one line at a time, you find that every comment is considered an executing line. We'd much prefer a single window with the executing statement printed at top or bottom of the screen. Programmers often get carried away by enthusiasm for fancy and faddish new approaches. So it is with two windows in this BASIC. They're a useless pain in the behind.

Error messages

Despite a large error window ("Syntax Error!"), and a red border around the line of error, Microsoft's error messages are no more useful now than they were five years ago. You aren't told where on the line the error is, nor why it's an error. Example: what's wrong with the statements below? You get an error message 'ELSE/ELSEIF/ENDIF WITHOUT an IF'. Is that the real problem? Those accustomed to systems with intelligent error handling and marking will weep. The problem? Shucks, we left out a **then**.

```
IF x OR y
PRINT "Whoa!"
END IF
```

Editing

The editor in Amiga BASIC is both primitive and clumsy. It is also very slow. The screen scrolls very slowly when you page up or down by screen page. It takes one full second to move the cursor from the top line to the bottom of a 19-line page unless you shift the cursor with the mouse. The interpreter of Amiga BASIC may be in assembler; the editor is done in molasses.

Editing within Amiga BASIC otherwise is fairly handy. You delete and copy lines or sections of code by highlighting the material with the mouse. You then copy the material to a buffer (called the 'clipboard'). You may copy erased code to a new position, shown by the cursor, with **Amiga P** (for **paste**). If you choose to **copy**, the original code remains, but a copy is put into the clipboard; you may insert that copy anywhere else in text. This works well on small segments. We tried to delete the second half of a larger program by highlighting with the mouse. By stopwatch it took over four minutes to highlight most of the material; then we crashed (we suspect that the clipboard buffer overflowed). So you must delete in small sections. In sum, you'll probably use this editor to debug code; it is far too slow and clumsy for long editing sessions.

Fortunately, Amiga's multitasking allowed me to use **Ed**, Amiga's screen editor, to write programs. And then, with **Ed** running concurrently, I crashed Amiga BASIC six times in two hours and forty minutes, with nary a program in memory longer than 80 lines. I kept getting 'Heap Full' errors.

von Lundsdorff

There's a text-to-speech demo on the Amiga BASIC disk. I cranked it up and found it hilarious. The default voice settings sound precisely like a Nazi villain, so I tried: "Let me introdusse myzelsf; I am Count Erich von Lundssdorff, und diss is Frankenstein." The extra sibilants were necessary for proper effect. The range of voices possible is limited: I tried for hours to get rid of the Teutonic flavour, without success. But those who want trolls, ogres, wizards and witches will have great fun. The built-in **translate$** function converts any Engish sentence into proper phonemes for the voice. If you want to bypass **translate$** and control the voice directly, you may do so — but with phonemes, glottal stops, and such.

Windows, mice and menus

Amiga BASIC gives the programmer control over all the features of the Amiga, from pull-down menus to windows and the use of the mouse and joysticks — if you care to use them. A very large part of the lanuage is devoted to these things; they are powerful and easy to use.

ML interfacing

Amiga BASIC provides both **varptr** (the address of numeric variables) and **sadd** (the address of a string), so that you may access any variable by its address. You may therefore pass these named variables to ML routines. You may also call ML library routines that you write yourself or those already present in Amiga.

A few nice touches

The index function to find a substring within a larger string (called **instr**) allows a secondary index, as in: **INSTR(7,X$,"Brown")**. In this case, you look for "Brown" within a larger X$, but you start searching at the 7th character in X$. The secondary index is optional. Very handy.

More importantly, you may identify variables either by defining their data type (as in Pascal or C), or by using suffixes, as in other BASICs. The identifiers shown below (% for a short integer, **&** for a long integer, and so on) take precedence over any stated definitions.

Definition	Meaning	Suffix form
DEFINT var	Short integer (16 bits)	var%
DEFLNG var	Long integer (32 bits)	var&
DEFSNG var	Real (7 digits, 4 bytes)	var!
DEFDBL var	Real (16 digits, 8 bytes)	var#
DEFSTR var	String	var$

In subprograms, the definitions may be purely local, or they may be **shared** if so declared within the subprogram. If they are shared, they become global variables in the main program, and may be used in other subprograms if declared there as **shared**.

Tests on variations

I ran a number of tests, and was surprised to find no difference in execution time with or without line numbers. Even more surprising: a program using defined variables (as above) runs faster than one where variables are suffixed with their type (**defint y** is faster than **y%**). Last, and most astonishing, multiple statements per line are slower to execute than single statements per line! In this language, there is utterly no excuse for cramming code together in the fewest possible lines. All these things are positive advantages: they reward simple, readable code.

Summary

This BASIC is much improved over ABasiC but just doesn't go far enough. The code is far easier to read and maintain than older, unindented, and unlabelled versions. It was hastily and drastically shortened at the last minute to let it fit in available memory. As time goes on and code is boiled down, or as more memory becomes available in the Amiga, I wouldn't be at all surprised to see **auto** variables, recursion, **reset**, additional optional data types, and some fully structured options. It is far superior to any Microsoft Basic I've seen before, and a large step in the right direction. □

Amiga BASIC sound and screen

by Chris Johnson

Amiga BASIC makes the advanced sound and graphics features of the Amiga easily accessible to the BASIC programmer. The use of 'bobs' (blitter objects) and sprites is fully supported in Amiga BASIC; screen images can be captured in an array and placed elsewhere on the screen or saved to disk; and the **sound** command makes music easy to implement. Unfortunately, some aspects of screen manipulation are not supported as fully as in ABasiC.

Using colour

Colours for the various graphic commands are defined using the **palette** command. Up to 32 colours can be defined by the amount of red, green and blue in the final colour. For example, **palette 1, .8, .6, .53** assigns a brown to colour-id 1. The number of colours accessible at any time is determined by the depth (number of bitplanes) of the screen (set by the **screen** command). If the **palette** command is omitted, Amiga BASIC uses the colours defined with the **Preferences** tool. **color <pen>,<background>** sets the background and foreground colours using the colour-id as defined with **palette**.

Graphics commands are executed using the pen (or other, specified) colour. These commands include:

- **Line**, which can also be used to draw a rectangle by specifying two opposite corners and adding **,b** after the coordinates or **,bf** to fill the rectangle with the pen colour;
- **Circle** can be used to draw arcs and any variety of ellipse;
- **Areafill** fills an area defined by **area** statements;
- **Paint** fills an enclosed area with a specified colour; and,
- **Pset** and **Preset** set a point on the screen. **Preset** differs from **Pset** in that, if you do not specify a colour, it uses the background colour.

Commands that draw lines or fill areas can be modified with **pattern**, which sets a 16-bit mask. **Pattern &hffff** would turn on all the bits when a line is drawn; **Pattern &hf0f0: line (0,0)-(100,100)** draws a dotted line. (The **&h** in these commands specifies that the following number is to be interpreted as hexadecimal.)

Bobs and sprites

Bobs and sprites are controlled with a series of **object** commands, each of which has a different suffix:

- **.Shape** assigns an object-id number to an object definition contained in a string expression;
- **.On** and **.off** make an object visible or invisible;
- **.X** and **.y** place the object at a specified point in the x or y axes or, used as a function, return its position;
- **.Start** and **.stop** set an object in motion or freeze it;
- **.Vx** and **.vy** set the velocity of the object in X and Y coordinates;
- **.Ax** and **.ay** set the acceleration in pixels per second per second;
- **.Priority** sets a bob's priority. A bob will appear in front of an object with a lower priority and behind one with a higher priority;
- **.Clip** defines an area of the screen beyond which an object will not be drawn;
- **.Hit** allows selection of which other objects will cause a collision;
- **.Close** releases memory assigned to an object.

Collisions between objects are detected by event trapping, which is enabled, disabled and suspended by **collision on, collision off** and **collision stop** respectively. When enabled, up to 16 collisions can be queued and read by **collision(<object-id>)**, which returns either the number of another object or a negative value indicating top, left, bottom or right border.

Object Editor

The creation of a bob or sprite can be done with a utility program called **ObjEdit**, which is one of the demo programs on the Amiga Extras disk. Written in Amiga BASIC, this program allows you to draw your bob or sprite on the screen with the mouse. It includes a 'zoom' command that enlarges the object for detailed work. The objects can be saved to disk and read into memory when necessary. Concise examples are given both in the manual (with a few errors) and on the disk.

Screen Get and Put

Areas of the screen can be stored in an array with **get**, then **put** to the screen. An array of sufficient size must first be dimensioned, then **get** is used with the pixel coordinates that define a rectangle containing the image. These can be **put** in one of several ways: **pset** or **preset**, or the images can be **and**ed, **or**ed or **xor**ed with the image already on the screen.

Printing to the screen

Printing to the screen can be done with **print**, **print using** and **write**. **Print** behaves in the normal fashion, as does **print using** for those who have used it before.

Write, on the other hand, prints an expression list, the items of which must be separated with commas. Numbers are printed without leading spaces, strings are printed in quotes, and the delimiting commas are printed.

Width defines the maximum number of characters that will print on a line: Amiga BASIC will not print a carriage return when it reaches the edge of the screen if **width** is set to the default of 255.

There is no **print at** command in Amiga BASIC (there is in ABasiC) and the **locate** command positions the cursor only by character, not by pixel. Positioning text at a precise location on the screen in combination with graphics is thus made somewhat difficult. The **ptab** command moves the cursor horizontally by pixel. There is no direct method for positioning text vertically on anything other than the normal printing lines.

There are two awkward ways around this limitation. The first is to print the text to the nearest line, then **scroll** a portion of the screen until it is where you want it. The second is to print the text somewhere where it will not interfere and store it in an array with a **get**. This can then be **put** anywhere. Neither of these is entirely satisfactory, and one misses the ease with which this operation could be performed in ABasiC.

For some uses, however, **scroll** is a handy command. Two corners of a rectangle are defined, followed by the number of pixels to scroll the screen in both the x and y axes. This allows diagonal as well as left, right, up and down scrolling.

There are also no commands for cursor movement, inserting characters, clear to end of display or clear to end of line. The ANSI standard commands which the system supports and which work in ABasiC (see Amiga Screen Magic, *TPUG Magazine* #22) do nothing in Amiga BASIC. Homing the cursor must be done either with **cls** or **locate 1,1**. Relative cursor movement (to the previous line, for example) has to be done by reading the current line with **csrlin**:

```
y = csrlin
x = pos(0)
locate y-1, x
```

Sound and music

The easiest method of getting a sound with Amiga BASIC is by using the standard **beep** (or **print chr$(7)**).

But you probably want to do more than this, and in Amiga BASIC it is not all that hard. After the convoluted method of producing sound in ABasiC, this is child's play. Unfortunately, there is no command for setting the ADSR envelope, as there is in ABasiC. ABasiC also contains functions that can tell you whether any voice is still playing. New sounds can then be sent to an inactive channel. Amiga BASIC has none of this.

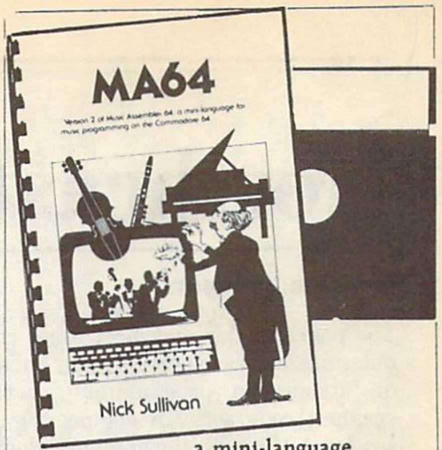
There are really only two commands you need to know to create music: **sound** and **wave**. Wave forms are set up in integer arrays of at least 256 elements; each element in the array must be in the range of -128 to 127. The array is then assigned to **wave** 0 to 3, corresponding to the four voices.

Notes are played with: **sound <frequency>, <duration>, <volume>, <voice>**; the last two parameters are optional. **Sound wait** suspends execution of **sound** statements until a **sound resume** is encountered, allowing synchronization of all four voices.

The music demonstration provided with Amiga BASIC, J.S. Bach's *Jesu Joy of Man's Desiring*, is a magnificent example of programming that will repay a close examination. It is not immediately apparent how the program works, but with a little effort it can be used as a driver for your own music simply by replacing the **data** statements.

Summary

Though Amiga BASIC is, on the whole, a great improvement over ABasiC, many features have been left out. The manual makes references to continued developments; let us hope that these are forthcoming. □


MA64
Version 2 of Music Assembler 64: a mini-language for music programming on the Commodore 64
Nick Sullivan

MA64 a mini-language for Music Programming on the Commodore 64

- Add music to your own programs
- Complete control of SID capabilities
- Fast, text-oriented music entry.
- Easy to learn, easier to use.
- Comprehensive Manual
- Dozens of powerful commands

29.95 CAN. 24.95 US

Ont. Residents add 7% Prov. Sales Tax
Cheque, Money Order, Visa or MasterCard
Postage & Handling Included

VISA MasterCard

WILANTA ARTS
WILANTA ARTS
6943 Barrisdale Dr.
Mississauga, Ont.
L5N 2H5
1-416-858-9298

Freeze Frame
WHAT YOU SEE IS WHAT YOU GET.

X10
POWERHOUSE INTERFACES WITH YOUR COMMODORE TO CONTROL YOUR HOME... FOR SECURITY

Double your Programming Power. Add speed and versatility to your C64 - Macro Assembler package.

Video anti-radiation shield with no-glare filter and grounding cords.

TOUCH BASE STATIC CONTROL WRIST REST
A. Grounding cord
B. Grounding aperture

- *Cartridge Expander for C-64*
- *Blitz compiler*
- *Flip & File*
- *Games Less 15%*

- *News Room and Clip Art*
- *Newsroom*
- *PaperClip*
- *Consultant*
- *Multiplan*

1535 Bayview Avenue
Toronto, Ontario
M4G 3B5
(416) 481-6807
Authorized Commodore Dealer

TPUG MEMBERS 15% DISCOUNT

Northcastle Structured BASIC

by Chris Johnson

The pages of this and most other computer magazines have seen many debates on structured programming versus spaghetti code, software engineering versus frontal assault, the imminent demise or triumph of BASIC as opposed to structured languages.

As with most controversies, the truth lies somewhere in between: if it were not possible to write a program without a carefully-drafted flowchart and copious comments, software libraries would be reduced to a fraction (hexadecimated?). On the other hand, if nobody introduced any structure into their software, few of the programs in those libraries would be more than two or three kilobytes long.

BASIC is often denigrated because of its lack of structure, but people use it and will continue to use it because it is both easy to learn and the easiest language to start with: most home computers are in a BASIC environment as soon as they are turned on. I have found that it is possible, even necessary with longer programs, to write in a structured fashion. In order to make structured programming easier, while remaining in a BASIC environment, I use a BASIC enhancement written by Mike Roche of Oshawa, Ontario, that is available from the TPUG library for the PET ((P)TD), C-64 ((C)MD), and the VIC 20. This useful utility is called Northcastle Structured BASIC.

Commodore BASIC does not make it easy to use indented listings. If you put spaces after a line number in a program, the editor removes them. Indented loops and structures make a program easier to understand. In Northcastle (which we'll shorten to STB for the purposes of this article), you get what you type:

```
10 for i = 1 to 9
20    a(i) = 5 * i
30 next i
```

Features

STB will run all programs written with Waterloo BASIC, and even has a few features missing from Waterloo.

IF-THEN-ELSE: Multiple line IF statements are supported:

```
110 if a = 1
120    b = a
130    c = a*b
140 elseif a = 2
150    b = a/2
160    c = b+a
170 else
180    b = a*2
190    c = a/b*2
200 endif
```

Structured loops: Structured programming usually avoids **goto** and other statements requiring the use of line numbers. Where a BASIC might use **goto 10** for an infinite loop, STB uses **loop . . . endloop:**

```
10 loop
20   rem (program in here)
30   rem ......
40 endloop
```

To provide an exit from the loop, **endloop** may be replaced with an **until** statement. That very common BASIC line:

```
10 get a$: if a$="" then 10
```

can be replaced with:

```
10 loop
20    get a$
30 until a$<>""
```

Since there is no reference to a line number, these lines could be placed anywhere in the program without alteration.

Structures can also be exited by the **quit** command. Improving on Waterloo BASIC, STB's **quit** can take a parameter: **quit(2)** will exit two levels of loops.

Until sets an exit condition for a loop; **while** sets an entry condition — the loop is not entered unless the condition is true. For example:

```
100 while a$<>chr$(13)
110    loop
120       get a$
130    until a$<>""
140    print a$
150    in$ = in$+a$
160 endloop
```

Procedures: Procedures are really named subroutines. A procedure is not executed in the normal sequence of a program; it must be **call**ed:

```
10 call print title
20 loop
30    call enter info
40 until in$ = ""
50 end
60 proc print title
70    print chr$(147);: rem
      clear screen
80    print "--------------"
90    print "Procedure Demo"
100   print "--------------"
110   print
120 endproc
130 :
140 proc enter info
150    rem (entry routine)
160 endproc
```

Procedures names can include almost anything but colons. I have even used **proc print run**.

Because there need be no reference to line numbers within a program, entire sections can be written separately and appended to form a finished program. (See the listing of **Appender**, a utility in BASIC that combines programs loaded from disk.) When the procedures are all collected, use a renumbering facility, such as that found in **BASIC Aid** (available from the TPUG library) or **Power** (from Pro-Line Software) or my own **Incrementer**, a simple line renumberer (invoked by **sys 49152**,x, where x is the amount for each successive line number to be incremented). **Incrementer** will not renumber **goto**s or **gosub**s, but there is no need for them when using STB.

STB differs from a true structured language in that there is no passing of parameters to a procedure, and the variables are not local to it. This means that care must be taken not to have conflicting variables when building a program out of prewritten procedures.

Restore: The **restore <line number>** is the only feature of Northcastle Structured BASIC that requires line numbers. Not that you are restricted to a number — an expression that resolves to a line number can be used instead.

It took me a while to reconcile the use of line numbers with **restore** and concatenating program segments — even using **Power** or **BASIC Aid**, a line number after **restore** will not be renumbered. There is an easy way around it, however.

Locations 57 and 58 on the VIC and C-64 (54 and 55 on the PET) contain the line number being executed in low-byte/high-byte format. Therefore, **peek(57) + 256*peek(58)** can be used after **restore**, assuming that the **data** statements you want to **read** are next after the current line. For example, I have on disk a set of procedures that I use in programs that include any date entry. The initialization procedure looks like this:

```
110 proc date init
120   restore peek(57)+256
      *peek(58)
130   while mn$(10)<>"Octob
      er"
140      dim mn$(12),ml(12),
         dy$(7)
150      for i=1 to 12
160         read mn$(i),ml(i)
170      next
180      for i=1 to 7
190         read dy$(i)
200      next
210   endloop
220 endproc
230 data "January", 31, "Fe
    bruary", 28, "March", 3
    1, "April", 30, "May",
    31
240 data "June", 30, "July"
    , 31, "August", 31, "Se
    ptember", 30
250 data "October", 31, "No
    vember", 30, "December"
    , 31
260 data "Sunday", "Monday"
    , "Tuesday", "Wednesday
    ", "Thursday", "Friday"
270  data "Saturday"
```

Renumbering this procedure after appending it to a program will not affect the execution at all.

In practice, I define a BASIC function to do the double **peek**:

```
def fn dp(x)=peek(x)+256*pe
ek(x+1)
```

Then, after **restore fn dp(57)**, a **read** statement will find the next **data** statement in the program.

Top-down programming: Perhap the greatest advantage of NorthCastle Structured BASIC is that it allows top-down programming. A database program could start like this:

```
 10 call initialize variable
    s
 20 loop
 30   call menu
 40   if u$ = "enter"
 50      call enter
 60   elseif u$ = "edit"
 70      call edit file
 80   elseif u$ = "load"
 90      call load file
100   elseif u$ = "save"
110      call save file
120   endif
130 until u$ = "quit"
140 end
150 :
160 proc menu
170    print "Menu": rem dum
       my proc
180 endproc
```

By creating dummy procedures at first, the logic of the program can be tested, and individual sections written one at a time.

State of the union

Northcastle Structured BASIC is small: less than 2,000 bytes. It relies heavily on the BASIC routines in ROM. An example of this is the **endproc** statement. I said earlier that STB procedures are just named subroutines. You can prove this by using **return** instead of **endproc**: nobody is any the wiser. I usually use **return** to make converting an STB program to Commodore BASIC a bit easier: leave the **proc** statement alone and **gosub** to the line after it.

STB can also co-exist with programmer's utilities, can be turned on and off at will, and can easily be ignored even when it is active; it does not prevent you from continuing to fabricate your favourite fettucini.

STB naturally slows down BASIC execution since it has to check for the added features, but if a routine really needs all the horsepower it can get, write it in CBM BASIC. Turn STB off when entering the routine and back on when finished.

Though not a full-fledged structured language, Northcastle Structured BASIC is an excellent way to start learning structured programming. It enables fast writing of code (I have written over 5K bytes in an evening) but does not interfere with the familiar BASIC environment.

```
 5 rem incrementer, chris
   johnson, 1986
10 fori= 49152 to i+ 94
   step 5 :forj=0to 4
11 reada:poke(i+j),a:c=c+a:
   nextj:read cs
12 ifc<>csthenprint"<2 down
   >Error in line";i:fori=1
   to25:a=abs(a-15):poke542
   96,a:next:stop
13 c=0:nexti
49152 data 32,241,183,142,
      93,691
49157 data 192,160,2,177,
      43,574
49162 data 141,91,192,200,
      177,801
49167 data 43,141,92,192,
      165,633
49172 data 43,133,251,198,
      251,876
49177 data 165,44,133,252,
      32,626
49182 data 86,192,201,0,
      240,719
49187 data 5,32,80,192,208,
      517
49192 data 251,32,80,192,
      32,587
49197 data 80,192,240,41,
      32,585
49202 data 80,192,24,173,
      91,560
49207 data 192,145,251,109,
      93,790
49212 data 192,141,91,192,
      32,648
49217 data 80,192,173,92,
      192,729
49222 data 145,251,105,0,
      141,642
49227 data 92,192,76,36,
      192,588
49232 data 230,251,208,2,
      230,921
49237 data 252,160,0,177,
      251,840
49242 data 96,0,0,2,138,236
```

```
 10 rem appender, chris john
    son, 1986 :
 30 rem set bottom of basic
 40 rem to top of program
 50 rem currently in memory
 60 t=peek(45)+256*peek(46)
    -2
 70 poke44,t/256
 80 poke43,t-256*peek(44)
100 rem clear keyboard
110 rem buffer and get nam
    e of file to append
120 poke198,0:input "<clr>
    <ctrl-n>File to append"
    ; fl$
140 rem If no file name
150 rem entered set pointer
    s back to normal
160 rem and end program.
170 if fl$=""then poke44,8:
    poke43,1: end
190 rem use dynamic keyboar
    d to load
200 rem new program, reset
    pointers
210 rem and reenter program
220 print"<clr>load"chr$(34
    )fl$chr$(34)",8"chr$(13
    )"<5 down>poke44,8:poke
    43,1:run
230 poke198,7:poke631,19:fo
    ri=1to7:poke631+i,13:ne
    xt:stop
```
□

COMAL: The Disk, The Cartridge

by Len Lindsay

COMAL is a language that combines the power of Pascal, complete with indented structures, with the ease and interactive nature of BASIC. The Commodore 64 versions include LOGO-compatible turtle graphics. Two versions are available for the C-64: one disk-based, the other cartridge-based.

A poll taken by Commodore itself reveals that COMAL is the first alternate language of choice (BASIC and machine language were more popular — but they come with the machine). There are two versions of COMAL for the Commodore 64, disk-loaded COMAL 0.14 (available from most Commodore user groups, including TPUG), and the COMAL 2.0 Cartridge. Both work in C-64 mode on the C-128.

The COMAL 2.0 Cartridge is the deluxe version of COMAL. It includes all the features of disk-loaded COMAL 0.14, and more. Some people consider it the best programming language on any 8-bit microcomputer. Once you use COMAL 0.14, you do not want to go back to BASIC. And once you use the COMAL 2.0 Cartridge, you will not want to go back to COMAL 0.14.

COMAL characteristics

With the cartridge, COMAL is there every time you turn on the computer. To go back to BASIC, you must type in the command: **BASIC**. To use COMAL 0.14, you must first load it from disk. The

First Column: COMAL 2.0
Second Column: COMAL 0.14
Third Column: BASIC 2.0

x	x	-	Understands TRUE and FALSE
x	x	-	DIV (integer division) operator
x	x	-	MOD (remainder) operator
x	x	-	Arrays with any minimum index
x	x	x	Integer numbers
x	x	-	Produce random integer in a range

Editing features

x	x	-	AUTO (automatic line numbers)
x	x	-	RENUM (renumber lines)
x	x	-	MERGE from disk
x	x	-	Syntax checking on entry
x	x	-	Delete blocks of lines
x	-	-	FIND and CHANGE commands
x	x	-	Pause a program listing
x	-	-	TRACE (debugging aid)
x	-	-	'Quote mode' disable/enable
x	-	-	Understands upper and lower case
x	-	-	Erase to end of line
x	-	-	Ooops key

Files

x	x	-	Binary sequential/random files
x	x	x	ASCII sequential/random files
x	x	-	Easy one-command random file use
x	-	x	GET from disk
x	-	-	Built in true ASCII conversion

Disk commands

x	x	-	CAT (catalogue of files on disk)
x	-	-	Pause catalog/send it to printer
x	x	-	STATUS (disk drive status)
x	-	-	COPY (copy files command)
x	x	-	DELETE (scratch files from disk)
x	-	-	MOUNT (initialize a disk)
x	-	-	RENAME a disk file
x	x	-	Knows when End Of File is reached
x	x	-	CHAIN one program to another

Numbers

x	-	-	Accepts Hex and Binary numbers
x	-	x	Includes Logical AND and OR
x	-	-	Includes Logical XOR
x	x	x	Includes trig functions

Input-output

x	x	-	TAB works on printer as on screen
x	x	-	Variable-sized print zones
x	x	-	Print zone same on printer as screen
x	-	-	Set up default printer types
x	-	-	Built in graphic screen dump
x	-	-	Built in text screen dump
x	x	-	PRINT USING formatted output
x	x	-	Select output: printer or screen
x	-	-	Select input: keyboard/batch file
x	-	-	INPUT AT and PRINT AT
x	-	-	Automatically-protected input fields
x	x	-	Allows null reply to input
x	x	-	Allows STOP key during input
x	x	-	Allows comma as part of input
x	-	-	User-definable character fonts

Structures

x	x	x	FOR loop
x	x	-	Integer FOR loop
x	x	-	REPEAT...UNTIL loop
x	x	-	WHILE...ENDWHILE loop
x	-	-	LOOP...EXIT loop
x	x	-	CASE structure
x	x	-	Multiple-line IF...THEN...ELSE
x	x	-	Call routines by name
x	-	-	External procedures and functions
x	x	-	Multiple-line procedures/functions
x	x	-	Parameters with procs/funcs
x	x	-	LOCAL or GLOBAL variables
x	-	-	ERROR HANDLER (trap errors)
x	x	-	Automatic indenting of structures

Sprites

x	x	-	Keywords for defining sprites
x	x	-	Keywords for setting sprite colour
x	x	-	Keyword for moving sprites
x	x	-	Built-in collision detection
x	-	-	STAMP a sprite image onto screen
x	-	-	Animate sprites, interrupt driven
x	-	-	Attach sprite shapes to programs

Graphics

x	x	-	Turtle graphics and X/Y graphics
x	x	-	Hi-res or multicolour graphics
x	x	-	Split screen (text/graphics)
x	x	-	Background/border colour keywords
x	x	-	Mix text and graphics on screen
x	-	-	Graphics text in any size
x	-	-	Graphics text sideways
x	-	-	Save a graphics screen to disk
x	-	-	Window capabilities
x	x	-	Line clipping within frame
x	-	-	ARC and CIRCLE commands
x	x	-	FILL command
x	x	-	PLOT a point

Sound

x	-	-	BELL command
x	-	-	Built-in sound commands
x	-	-	Control sound envelope
x	-	-	Interrupt-driven music

Machine language

x	x	x	Call machine-code routines
x	-	-	Call machine code by name
x	-	-	Link machine code to programs
x	-	-	Parameter passing to ML routines

Miscellaneous

x	-	-	Modem-communications built in
x	x	-	Function keys defined
x	-	-	Function keys alterable by user
x	x	-	Stop key disable/enable
x	-	-	Cursor command
x	x	-	No 'garbage collection'
x	-	-	Joystick/paddle/lightpen keywords
x	x	-	IN (built-in string search)
x	-	-	Store a text screen for later use
x	x	-	Long variable names
x	-	-	Can sense SRQ interrupt
x	x	-	Can change part of a string
x	-	-	Built-in clear screen command
x	x	x	PEEK, POKE, SYS, GOTO

fastload version now available loads in less than 20 seconds, so the wait is not long.

Disk-based COMAL is a public domain program. You are allowed to give copies of COMAL 0.14 to others. All COMAL 0.14 programs can therefore be run on any Commodore 64 or 128. Just include the COMAL 0.14 system on the disk with your programs. COMAL 2.0 programs will not run without the cartridge: only people with COMAL 2.0 Cartridges may run COMAL 2.0 programs.

A program written in COMAL 0.14 can run under COMAL 2.0. To transfer a COMAL 0.14 program to 2.0 you must **list** the program to disk, then **enter** it into 2.0. A program that is saved to disk is tokenized, and COMAL 2.0 and 0.14 use different tokens.

COMAL 0.14 and 2.0 are similar in many ways. This is to be expected, since COMAL is a standardized language. They both adhere to the COMAL Kernal, though COMAL 0.14 is missing a couple things, such as **val** and **str$**. The Cartridge includes the complete COMAL Kernal.

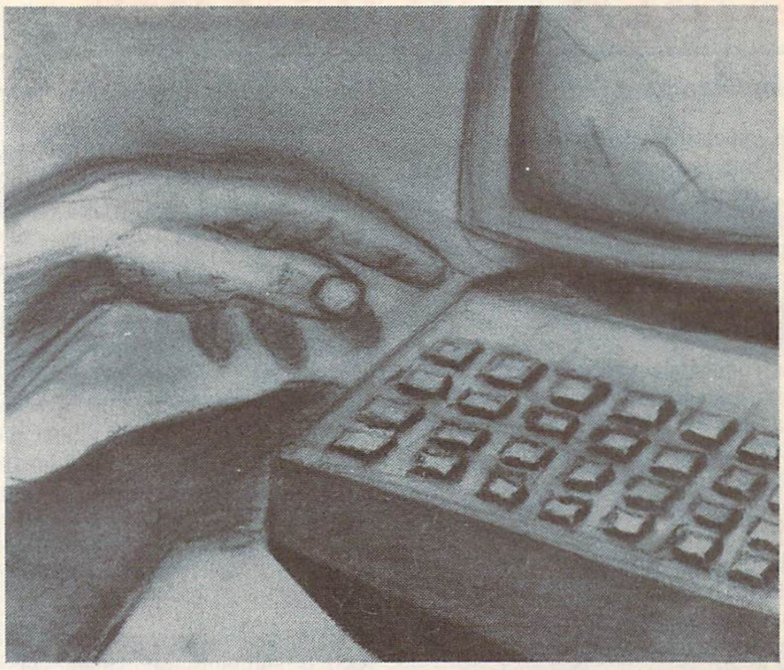
One of the main differences is with graphics and sprites. COMAL 2.0 requires parentheses around parameters while 0.14 does not. Also, **closed** procedures are included in both versions, but in COMAL 2.0 they are a bit more closed: you must **import** any procedure or function names that you call from within a closed module.

The COMAL 2.0 Cartridge adds many enhancements. Remember, it is a 64K system! Program editing is improved. **Auto** and **renum** are retained from the 0.14 version, but many more features have been added. A procedure can be listed by name — no need to remember what its line numbers are. You can even have it listed without line numbers using the **display** command. COMAL 2.0 automatically lists the COMAL keywords in upper case and your variable names in lower case. You may use either upper or lower case when you type your lines in.

Like a good word processor, the Cartridge includes **find** and **change** commands. The **change** is selective: you don't have to change every occurence of the letters you specify.

Perhaps the fastest way to compare the two versions of COMAL is with a chart. The chart of features in the box accompanying this article is subdivided by categories. BASIC is also included on the chart, even though it lacks most of the added features. An **x** in the column means the feature is included. A **-** in the column means it is not included. □

TPUG PROGRAMMING CONTEST



TPUG is once again offering you the opportunity to reduce the costs of your hobby. The Librarians Committee of TPUG is sponsoring a programming contest as a means to encourage you to submit your programs to the library. The winner of this contest will be selected at random from the names of the submitters of all programs accepted by the librarians from the submissions received between the first publication date of this notice and Friday, October 31, 1986. The more programs you submit, the greater your chance of winning.

RULES

- Submissions must be received on or before the deadline.
- Submissions must be on diskette (VIC programs may be submitted on cassette — two copies, please).
- Submissions must be original material.
- Submissions can be for any Commodore machine.
- Submissions should indicate that they are contest submissions.
- All submissions become the property of TPUG.
- TPUG general policy of returning a disk of your choice on acceptance remains in effect for all submissions.
- Unaccepted disks will be returned.
- Freeware submissions will not be accepted for contest consideration.
- Submitter's name must be included in a comment statement at the start of the program as well as on the front of the disk.
- First, second and third prizes will be awarded consisting of 100, 50, 25 blank disks respectively or 25, 10, 5 disks (respectively) from the TPUG libraries.

The Librarians Committee

Amiga Dispatches

by Tim Grantham

I keep expecting the excitement to die down. But as each new productivity, game, creativity program for the Amiga is announced; as each new peripheral is introduced; as each new company declares its intention to develop products for the Amiga, I am amazed anew by the potential of this machine.

Now come the intriguing rumours of the next generation of the Amiga, code-named 'Ranger'. This will apparently be a 68020-based computer with 1024 by 1024 pixel graphics and complete compatibility with all Amiga hardware and software. There is even speculation that it will be released this fall, at a cost of between three and four thousand dollars (US). C-A (Commodore-Amiga) has announced a revision of the expansion bus on the Amiga to accommodate the Ranger: as I understand it, all future Amigas will have a 100-pin bus instead of the current 86-pin bus. C-A is working with hardware developers to make existing peripherals upgradable to the new standard.

Meanwhile, there's lots happening in the here and now...

Software news

Soft Circuits released **PCLO**, their PCB CAD program, on April 2. Retailing for about $1000 (US), it has features superior to those of similar programs available for PC clones at twice the price. Upgrades and additional products to come will include autorouting, schematic capture, and circuit simulation... Borland has decided *not* to develop any products for the Amiga. Many Amiga owners had eagerly awaited the appearance of **Turbo Pascal**. Ashton-Tate has also declined to port any of their products, although spokesman Steve Silverwood has noted that they *will* be publishing **dBASE III** on 3.5 inch disks for the new IBM machines. These will most likely run on the Amiga under the **Transformer** PC emulator... It is rumoured that Microsoft will be adapting all of their major products to run under OS-9, which means that these products will be able to run on all the 68000 machines, including the Amiga, once TLM Systems has finished porting OS-9... Speaking of operating systems, there are persistent reports of successful efforts to port Unix to the Amiga.

Those who have seen the impressive jet simulator demo program currently screaming around the public domains will be interested to know that a release version is nearly ready, complete with aircraft carriers, airports and additional scenery... Mindscape's **The Halley Project** for the Amiga has been published. Word is that it is a dramatic improvement on the previous versions, with superb graphics and sound. I hope to have a review of this product in next month's issue... A hint to players of **Arctic Fox**: shoot down the air convertors as quickly as possible. Besides being worth a lot of points, destroying them preserves oxygen and gives tank commanders more time to get to the main fort.

Even more languages are appearing: CSI is shipping the Amiga version of Multi-Forth, in hot pursuit of MVP Forth... ABSOFT will be shipping an AmigaBasic compiler in June/July for a measly $295 (US)... Steve Jusik is porting his MacNosy debugger... The assembler that comes with the highly-rated Aztec C compiler does not, unfortunately, produce code that is compatible with the Amiga Assembler marketed by C-A. Aztec has promised that such compatibility will be part of their first update. Meanwhile, you can get around the problem by writing a program to translate the non-standard 68000 pseudo-ops used by the Aztec assembler to the standard ones used by the Amiga Assembler...

You may remember that v1.1 of the Amiga OS was written to provide compatibility with the 68010 and 68020 upgrade chips to the 68000 CPU. Unfortunately, none of the current math libraries in the OS, including IEEE and FFP, support the 68881 math coprocessor chip, though Randy Weiner of CBM has stated that they can be rewritten to do so. Randy has also mentioned that a much-needed upgrade to **TextCraft** is in the works; and that C-A will soon be marketing **FontEditor**, something that will be of great interest to the many people looking at the machine as a possible character generator for broadcast purposes. Randy did *not* mention that version 1.0 of Amiga Assembler was shipped with a couple of major bugs, first publicly spotted by Jez San of Argonaut Software in England, that render the include files useless: first, the lack of the **dos lib.i** file, which means no DOS operations can be carried out; second, and more important, a major error in the **exec/funcdef.i** file, which means that all Exec functions will be doomed to failure. San says the fix consists of locating the line in **funcdef.i** that reads:

```
func cnt     set    4*-6
```

and changing it to read:

```
func cnt     set    -30.
```

The widely-advertised **Gizmos** package is getting an excellent response from users. The fifteen utility programs include a fine, fast VT100 terminal emulator, that some say is worth the $50 (US) alone. C-A's **AmigaTerm**, bundled with the Amiga modem, also provides full VT100 emulation... Speaking of terminals, a couple of bugs have finally crawled out of MSS's **Online!** terminal program. It seems that if the capture buffer overflows, it will truncate the file rather than write the overflow to the disk. You can cope with this either by opening the capture buffer to a size greater than that of the expected file, or by doing as MSS suggests: setting the size of the buffer to 1, which will cause the file to be streamed to disk as it comes in. Another problem with **Online!** is that it does not trap **ctrl-n** and **ctrl-o**. These two characters will switch the alternate character set in and out, right when you are trying to read the message about the ultimate meaning of life. This can be solved by screening them out with a look-up table provided in the program... Aegis Development's **Images** program will not print with the correct aspect ratio on the JX-80 and other printers. Aegis has promised that this will be fixed in the free-to-owners upgrades... EA's (Electronic Arts) **Financial Cookbook** will not work with a 68010 chip installed...

Some encouraging announcements of music software have been made recently. Activision will soon be releasing **Music Studio** for the Amiga. Features apparently include complete musical notation on screen, and MIDI capability with control and scoring of up to 15 channels. Magnetic Music will be marketing Cherry Lane's **Texture** composition tool. In addition, they have announced a $120 (US) card to connect the Roland

MPU-401 MIDI controller to the Amiga; and a Yamaha DX-7 voice editor/ librarian...

Some other products announced... The VIP **Lotus**-clone is now shipping at $199 (US)... SSI Software are working on an Amiga version of **WordPerfect**... Charlie Heath has completed **TxED** and is selling it for $39.95 (US)...

EA has finally released an official policy on copy protection. All of the Deluxe series of creativity/productivity programs will use a key-disk protection scheme, allowing unlimited backups (minus the key) to microfloppy and hard disk drives. The key-disk need only be present at boot-up. A completely unprotected version can be had for $20 (US) by mailing in the warranty card. Those who bought the protected backups can obtain an unprotected version free of charge by mailing in the protected copy. Although none of this applies to EA's games, it should pacify those who were literally organizing a boycott of EA's products.

Hardware News

Lots of printer information this month... Epson is apparently discontinuing their JX-80 colour printer. However, look out for a colour printer from Canon for somewhere between $200 and $500 dollars (US), complete with Amiga printer driver. This machine will reportedly deliver very high-quality graphics that can cover the full range of Amiga colours... Redmond Cable is selling a cable and driver to permit the Amiga to use the Apple Imagewriter. Call (206) 868-2168 for details... There is support for the HP Laserjet laser printer in **Preferences**, but one can't access the 300 dots-per-inch resolution. Randy Weiner posted a fix: "You need some way of reading the specific driver file... (AmigaBasic works nicely). Read through the files to byte $0F2E, it should be set to 0200 (obviously I mean 'word $0F2E'). And, at offset $0F3C should be the word 0300. These offsets represent the settings for default and letter quality. Use the following table to set these values as you desire:

0100:	75 dpi
0200:	100 dpi
0300:	150 dpi
0400:	300 dpi.

That's all there is to it."

Look for a review soon of Comspec Communications 2 Megagbyte RAM board for the Amiga. It weighs in at a hefty $1450 (Cdn.) and is shipping now... The WCS (Writable Control Store) is part of the motherboard on new units, rather than being placed on a separate daughter board. Some are proposing that EEPROMs (Electronically Erasable Programmable Read-Only Memories) be used for the WCS to enable the machine to boot on power-up without the Kickstart disk, for such applications as bulletin boards. The EEPROMs would still provide the ability to upgrade the operating system... SoftCircuits has decided to go ahead with production of a small card that will interface any standard 40-track or 80-track 5.25 inch disk drive to the Amiga. It will cost about $50 (US) and be ready on or about May 1... The **Futuresound** audio digitizer mentioned in last month's column sells for $175 (US), and includes recorder, cables, microphone, and software. Call (617) 488-3602 for details... A-Squared's video digitizer **LIVE!** can now produce HAM (Hold And Modify) pictures that can display all 4096 colours available...

MicroForge, the first maker of hard-drives and expansion boards for the Amiga, have announced that they will be supporting the new 100-pin expansion bus on the Amiga. The new expansion boards will contain both 86-pin and 100-pin slots, and all other products will be available in either size. Owners of current 86-pin only expansion boards will be able to obtain upgrade kits. MicroForge has also announced a tape backup drive that should be available April 18. For those who want to supply their own hard drive, MicroForge will also sell their SCSI interface card, dual controller, and configuration software separately. Call (404) 688-9464 for more info...

The Transformer

C-A is at last releasing the **Transformer** IBM PC emulation software with the Amiga 5.25 inch disk drive. There are conflicting reports of its price, with some dealers selling the disk drive and **Transformer v1.0** as a package for $249.95 (US), while C-A says that the list price is $399.95 (US), or $299.95 for the drive and $99.95 for the **Transformer**. The program does not allow multitasking, nor use of those programs that generate colour graphics. It will run most programs written for both PCDOS 2.1 and higher, and most flavours of MSDOS, though some have reported problems with MSDOS v2.11. It will operate with any drive connected to the Amiga, so if you can get MSDOS software that is already on 3.5 inch disks, such as those for the Kaypro 2000, you will be able to get by without the 5.25 inch disk drive. (The **Transformer** does appear to have problems with the second 40 tracks on the Data General 3.5 inch disks.) Extensive beta-testing has shown that most programs run at between 70% to 85% of PC speed, except for some telecommunications programs, which are drastically slowed. William Harris reports that he was able to use **Copy II PC** to copy **Lotus 1-2-3** to the Amiga's 3.5 inch drives and run it without the original disk.

A company called Softeam, Inc. are promising their own version of the Transformer for release the third week of April. They claim it will provide 100 per cent compatibility, colour graphics support, completely free use of AmigaDOS formatted disks, and multitasking, all for $70 (US).

Some are claiming to have seen a demonstration from C-A of the hardware accelerator for the **Transformer**. Apparently, this unit will provide 100% emulation, including colour graphics, the speed of an AT, multitasking with Amgia-DOS and the ability to address up to 2 Megabytes of additional RAM. It will cost in the neighbourhood of $100 (US) and could be available as early as June.

Blits and pieces

Addison-Wesley has delayed the publication of the complete *ROM Kernel Manuals* yet again to June. Others have leapt into the breach, though the hurry shows in some of them. The *AmigaDOS Reference Guide* from Compute! Books, for example, repeatedly reverses the functions of control-O and control-N in describing the console device. Likewise, *The Amiga Programmer's Reference Guide* (also from Compute!) devotes a whole chapter to a new language called 'BACIC'. A quick look at both books indicated that, typos aside, they are of Compute!'s usual high calibre. Sybex and Bantam also have published books on Amiga internals. Look for reviews of these books in upcoming issues of *TPUG Magazine*.

Look for guest appearances of your favourite computer on *Miami Vice*.

I'll close with a couple of hints from Randy Weiner, Amiga engineer at CBM, West Chester: you can reduce flicker in interlace mode by using non-contrasting colours and by viewing the screen under tungsten rather than fluorescent lights. Also, there may be some disk glitches caused by interference from the magnetic fields of the adjacent 1080 monitor. Try placing a sheet of aluminum foil between the monitor and the CPU to act as a shield. □

ESCape G 2

by Adam Herst

Without software a computer is just a door stop (two computers can be put to use as bookends). This month I'm going to focus on the software that is slowly percolating its way into the market, both for the 128 mode and the CP/M mode.

Software strategies

The only truly innovative aspect of the C-128 is the inclusion of three distinct modes of operation, each capable of running radically different types of software. The rationale behind this was to provide a large, established source of software for the computer on its release.

This end has been achieved admirably. The C-128 was a productive machine from the minute the first one rolled off the assembly line. Software for the 64 and CP/M modes was immediately available. At the magazine, our C-128 was pressed into service from day one, running word processor, terminal and BBS programs, and hooked up to all sorts of standard and non-standard peripherals. In contrast, the Amiga, while occupying an exalted place on a desk of its own, is used only for the occasional download and the constant display of abstract designs courtesy of **Deluxe Paint**. This is not to belittle the Amiga, one of the most powerful and promising of personal computers, but to illustrate the success of Commodore's C-128 design strategy.

Nonetheless, this success must be qualified. The 'modular' ability of the C-128 has created its own problems related to software: misleading, if not false, advertising; a dearth of 128-mode software; and instances where CP/M software costs two or three times the C-128 system itself.

As the long awaited upgrade to the very successful C-64, the C-128 is able to run all of the C-64 software in its 64 mode. Documented problems appear to be with 1571 emulation of the 1541 drive, not with 64 mode itself. This has meant that the vast quantities of public domain and commercial programs run as well, if not better, on the C-128. This has given rise to an approach to software marketing which I consider questionable at best: the common practice of advertising C-64 software as software for the C-128. This software runs only in 64 mode on the C-128 and, in the case of cartridges, can inhibit the ability to access the other two modes of operation. This can only lead to confusion and bad feelings on the part of the user. While it can be argued that this information is directed towards the novice user, unfamiliar with computing, I would think that even the most ignorant of C-128 users knows one thing: *the computer can run C-64 software!* The designation *C-128 software* should logically be given only to programs that run in 128 mode.

If this were to happen, you would see a substantial reduction in the number of ads for C-128 software. This is due to the excruciatingly slow release of dedicated 128-mode software. The reason for this is purely financial. Early impressions of the C-128 were that it was merely a 128K 64. Since the installed market of C-64s is so large, and the 128 can run all 64 software, it seemed more economical to manufacturers to continue writing 64 software, thereby tapping both the C-64 and C-128 markets. Familiarity has bred respect however, and the power of the C-128 is being recognized. The advanced capabilities of the C-128 (80 columns, fast serial transfer, 2MHz clock speed) demand new and improved software. Manufacturers are recognizing this, and the first generation of 128-mode software is beginning to appear, with a power and sophistication unattainable in 64 mode. The tide is beginning to turn!

This slow development time was not unforeseen by the designers at Commodore. CP/M mode software was supposed to fill this gap with the megabytes of tried and true CP/M software available for practically any application. Well-intended as it was, this idea had a large ingredient of fantasy. While a large amount of CP/M software was available even a few short years ago, the supply is rapidly drying up. This is not to say that the CP/M operating system is dead, just that it receives little commercial support. What software *is* available can cost two or three times the cost of the C-128 system it is to be run on. At last look, **Wordstar Professional** was $499 Canadian, and **$Man**, a popular accounting package, listed at $1200! While I have managed to install both programs on the C-128, their cost will likely prohibit their widespread use in the Commodore world. This fact has not escaped manufacturers. With over 500,000 C-128s sold, and the number growing all the time, a lucrative new market now exists for what are probably discontinued or de-emphasized products. Re-releases of these products at reduced prices are no doubt at hand.

These observations have led to the establishment of a *TPUG Magazine* policy on reviewing CP/M software. First and foremost, the program must run on the C-128. Secondly, we feel that software priced at over $200 does not reflect the budget or interests of the average C-128 user, and will not normally be reviewed. Look for reviews of CP/M products that do meet these criteria.

Mice-capades

One of the newest peripherals for the C-128 is the 1350 mouse. A quick glimpse of this rodent was provided to me by Computers for Less in Toronto. A difficult product to evaluate on its own, the absence of menu-driven C-128 software made it nearly impossible. Physically attractive, it is a solid, well-built unit that is externally identical to the Amiga mouse, apart from a slightly remodelled connector. This similarity prompted a quick try on the Amiga with no success. The 1350 mouse does work on a C-64, though. A quick test with **Doodle** showed it to be very responsive with a fine discrimination of movement. Tests with other programs requiring a joystick also worked, with varying degrees of success. It seems, therefore, that the mouse is like an upside-down trackball, and thus should even work with a VIC 20! With the right software, this could be a very nice addition to your system.

Overdrives

While most people have purchased the 1571 disk drive to use with the 128, the 1541 still functions in all three modes and can be a useful second drive for backup purposes. A recent tip travelling around the BBSs is to access a unique feature of the 1541 to achieve an increase of about 20 per cent in the data transfer rate. If you have used a 1541 with a VIC 20, you are probably aware that issuing the **UI-** command to the disk drive on channel 15 will increase the data transfer speed to take advantage of the VIC 20's slightly less finicky timing. This capability can also be utilized in the fast mode of the

C-128, and with a greater gain of speed than on the VIC.

To implement this on the C-128 a number of points must be kept in mind. The 128 mode is faster than the VIC only in fast mode (when in slow mode it is slower than the C-64). When fast mode is activated, the 40 column screen is blanked. If you do not blank the 40 column screen, a faster data transfer rate will only succeed in locking up the system. To avoid this problem, issue the **fast** command at the beginning of the session (and after any resets) and enjoy a slightly faster 1541.

This increase in speed can also be obtained in CP/M mode, with a few more restrictions. The problem is in shutting off the 40 column screen and issuing the **UI-** command. While it is possible to issue these commands in 128 mode, then soft-boot CP/M, the act of booting resets the 40 column screen. This will result in a lockup when the 1541 is first accessed. If you are booting off a 1541, the system will lockup before CP/M is even booted! Since it is not possible to issue the **UI-** command from within CP/M this precludes using the faster speeds if all you have is a 1541.

If you are using both a 1541 and a 1571, however, you *can* use the faster speed. From 128 mode issue the **UI-** command. Then soft-boot CP/M from 128 mode, and turn off the 40 column screen with **conf.com** (available on TPUG disk (Z)AA) to avoid locking up the system. This procedure increases CP/M 1541 access by 23 per cent!

If you are the owner of two 1571s, the last few paragraphs on 'obsolete' equipment may have seemed irrelevant to you. Well, you ain't heard nuthin' yet! It seems that a few C-128s have been relased with Datasette access problems. A TPUG member in Minnesota has informed me that he has gone through three C-128s, all with the same problem, the most current having a serial number CA1846221. Since that conversation I have tried Datasettes with three C-128's, all successfully. If you have a datasette, give it a try on your 128. If you are not successful, drop me a line with the serial number of your machine. While this is not likely to be a major problem for the majority of users, the 128 was advertised as working with the Datasette and consequently should do so. Documented evidence to the contrary may produce some appropriate result from CBM.

Finally, a word for the 1571 owners. Sensitivity to monitor and TV interference was a major problem for 1541 users. This has not been alleviated with the 1571. If anything, the problem seems to be heightened. A number of users have reported that the drive must be moved farther from the offending source when the computer is in CP/M mode. With limited desk space, restrictions like that become a real headache.

Upgrade upgrade

Don't let these complaints give you the wrong impression, I'm still enamoured with the machine, and can't keep up with all the information that is pouring in. Its almost as hard keeping up with all the revisions to the CP/M Plus system for the C-128. It seems that the latest release, dated Dec. 6, 1985, contains a small bug. You won't have noticed it unless you are using a 1525/801 or compatible printer. This bug causes the printer buffer to eat the last line of text instead of dumping it to the printer. The release that corrects this bug is dated Dec. 8, 1985, but has not been officially released. This, however, is the version being packed with the new C-128s.

World of wonders

We end up this month with one of what I hope will be many puzzles, tips and discoveries from Jim Butterfield. Jim has informed us of a mysterious machine-code routine which magically appears in high RAM of bank 0. Look for this routine at address $ffd0 to $ffee. This routine appears on power-up but is *not* copied from ROM. Jim's challenge is to identify the source of this routine. If you think you can do it, send your response into *TPUG Magazine* care of myself. We'll let Jim pick the correct response since even we don't know the answer.

That's it for now; stay tuned for more flashes as they come in. And remember, 'Appreciate your 128!' □

TPUG BBS

The NEW telephone number is:

(416) 273-6300

Operating hours:
24 hours per day
7 days per week
The password is . . .

CONCEPT

Marketplace

COMPUTER RENTALS

We buy, trade, sell and repair Commodore computers

• 1541 Alignment	$ 29.00
• 4040 or 8050 Drive	$ 895.00
• 2031 Drive	$ 325.00
• Superpet	$ 895.00
• Education Software	$ 2.00
• Buscard	$ 99.00
• Paperclip	$ 66.00
• Consultant	$ 99.00

17 Cedarbank Crescent
Don Mills, Ontario M3B 3A3
Tel: (416) 446-1035

Spain • France
England • Germany

History from the beginning of civilization for any country, Europe and the Middle East

$29.95 incl. everything
Disks are full, 14 blocks open

Attn: World History
(416) 547-3086

273 Britania Ave.
Hamilton, Ontario
L8A 1Y9

C-64, C-128/64 Mode

This space could be advertising *your* product.

Call John Matheson, (416) 445-4524

A Layman's Guide to Burst Mode

by M. Garamszeghy

Part 1: Command Summary

The 1571 disk drive is one of the most versatile mass storage devices available for Commodore computers. Its disk operating system (DOS) supports an extended set of commands that allows the drive to, among other things, create, read and write disks in a wide variety of formats with relative ease. The description of these commands — collectively called 'burst mode' — in the 1571 instruction manual, is at best cryptic and, in most cases, downright confusing. This article is part one in a series designed to demystify this extremely powerful and useful set of commands. In this month's article, I will present a brief description of the syntax and function of each of the burst mode commands. In future instalments, I shall examine the operation and structure of the high-speed data transfer (burst) protocol used by many of the burst mode commands.

Many, but not all, of the burst mode commands have analogous commands in standard Commodore DOS. Why then is there a need to duplicate them in burst mode? The answer is speed. The data transfer rate of a 1541 drive (or a 1571 in 1541 mode) is about 350 bytes per second. The 1571 in fast mode is about 1200 bytes per second. With burst mode, data can be read or written at the blistering rate of up to 3800 bytes per second! In addition, burst mode allows you to access disks formatted in the non-Commodore industry standard, MFM (maximum (or modified) frequency modulation) format as well as Commodore GCR (group coded recording) format.

Burst mode commands are accessed by sending a specific set of characters through the disk command channel, in a manner similar to the usual Commodore DOS commands such as **n0:**, **s0:**, **b-p:**. The first two characters of all burst mode commands are **u0**. This allows burst mode to be accessed by a BASIC statement beginning with **open 15,8,15,"u0"**. What comes next depends on the desired function and its options. The first additional character selects the function and, perhaps, one or two primary options. The remaining characters select secondary options.

Before data can be read from, or written to, a disk using burst mode, it is necessary to *log in* the disk. After logging in, data can be read or written at will. If you remove the disk and change it, or even re-insert the same one, you will have to re-log the disk. The disk can be logged in by one of several methods. The simplest is to use the *inquire disk* command. This command will return a single status byte containing data on the current disk format and error status. If more detailed information is required about the format of the disk (such as number of sectors per track, sector numbering system, et cetera), then the *query disk format* command should be used. This latter command can be used to analyse the format of a specific track on the disk, while *inquire disk* only looks at the first track on the disk.

Data can be read from the disk by one of two methods. The first method is the *fast load* command. Unlike the other burst mode commands, *fast load* acts on an entire file and will only work on files stored in normal Commodore DOS GCR format. The routine will work with either PRG or SEQ type files. For consistent performance, the DOS wild card character should be appended to the filename being read. (For some quirky reason, *fast load* will not always recognize a legitimate filename unless it ends with a *.) The second method, *read specific sector*, is similar to the Commodore DOS **Block-Read** command. This command will work with either GCR or MFM disks. Unlike the DOS **Block-Read**, the burst mode command can be used to read more than one sector at a time. The order in which multiple sectors are read can be changed using the *set sector interleave* command. (Note that this is *not* the same as the sector interleave sub-command of the *utility* command.)

The default interleave of 1 corresponds to contiguous sectors. That is, if you started at track 10, sector 1 and read 3 sectors, you would read sector 1, then 2, then 3. If you changed the interleave to 3, you would read sector 1, then 4, then 7. Commodore DOS has, in most cases, an interleave of 10. C-128 CP/M has an interleave of 5, while most MFM disks do not use software interleaves. If you have ever traced a file through its various tracks and sectors using the 1541 **Display Track & Sector** program, you will see that in a file occupying consecutively allocated sectors, the sector numbers jump by 10 each time. (An exception is the directory file, which has an inteleave of 3.) Similarly, if you note the disk status display in the lower right corner of the CP/M screen display, the sector indicator jumps in increments of 5 each time a new sector is accessed. Multiple sector block-reads do not appear to be very useful unless you know that the data you want to read occupies consecutive sectors. One application of this function would be in a high speed disk copier, where a large number of sectors are read and then written to the corresponding locations on a new disk. The number of bytes transferred per sector read is equal to the number of bytes per sector, plus one. Therefore, 129, 257, 513 or 1025 bytes will transferred, depending on the sector size. The first byte is a status byte, followed by the data bytes.

Analogous to the *read* command is the burst mode *write specific sector* command. This allows you to write one or more sectors to a specific location on a GCR or an MFM disk. This command, however, is slightly more complicated than the read command because both burst data input and output are required. The calling program sends one sector of data, then waits for the drive to return a status byte. If more than one sector is to be transferred in sequence, then the next sector can only be sent once the status byte from the preceding sector has been read from the burst data channel.

The 1571 drive is capable of formatting disks in a variety of different types. The use of the MFM formatting procedure was described in a previous article (Formatting MFM Disks, *TPUG Magazine*, March 1986, page 32). Burst mode GCR formatting does not create a disk directory or BAM sectors. Its usefulness lies in the creation of custom disk formats with unique directory sectors and BAM (or functionally equivalent indexes). It can also be used as a high speed formatter for a disk copying program.

The final set of commands allow you to change various DOS parameters. These are not really burst mode commands but, since they are included in the burst mode chapter in the 1571 manual, a brief description is in order. The sector in-

1571 Burst Mode Commands

Function	Byte Sequence "u0" + chr$( )	Burst Input	Burst Output
Read a specific sector			
MFM disk side 0 or GCR disk (either side)	64, track#, sector#, # of sectors (usually 1)	none	for each sector read: one status byte then data bytes
MFM disk side 1	80, track#, sector#, # of sectors	none	as above
Write a specific sector			
MFM disk side 0 or GCR disk (either side)	66, track#, sector#, # of sectors	data bytes	one status byte after each sector transferred
MFM disk side 1	82, track#, sector#, # of sectors	data bytes	as above
Inquire disk: reset drive and log in MFM or GCR disk before a read or write			
MFM disk side 0 or GCR disk	4	none	one status byte
MFM disk side 1	20	none	as above
Format disk			
MFM single-sided	70, 129,0,sector size (0 = 128 bytes/sector, 1 = 256, 2 = 512, 3 = 1024), last track# (default 39), # of sectors per track, starting track# (default 0), track offset (default 0), fill byte (default hex e5)	none	none
MFM double-sided	first byte = 102 then as above		
GCR disk double-sided, no directory or BAM	6,0,ID byte#1, ID byte#2	none	none
Set sector interleave (for multisector read and write)			
Set interleave	8, interleave	none	none
Read last setting	136	none	last setting
Query disk format: analyze disk format (GCR or MFM-sector size, sectors/track)			
Side 0, track 0	10	none	one status byte then: nothing else if GCR disk or unreadable format, or else: another status byte, number of sectors on track, logical track#, min sector#, max sector#, hard sector interleave
Side 1, track 0	26	none	as above
Side 0, track n	138,n	none	as above
Side 1, track n	154,n	none	as above
Inquire status: check drive status or load status register			
Log in disk with new status	76, new status	none	none
Check last status	140	none	status from last I/O
Check if disk was logged	204	none	old status if logged or status error code 13
Set utilities			
Sector interleave	62, 83, value	none	none
# of retries on errors	62, 82, value	none	none
ROM signature analysis	62, 84	none	disk LED blinks 4 times if analysis fails
Mode select	62, 77, mode (mode = 48 for 1541 mode or 49 for 1571 mode)	none	none
Side select	62, 72, head (head = 48 for side 0 or 49 for side 1)	none	none
Device # change	62, dev# (dev# = 4 to 30, normally 8 to 11)	none	none
Fastload: read an entire GCR file			
SEQ file	159, filename character bytes . . . + "*"	none	for each sector read: one status byte then 254 data bytes; for last sector: status byte = 31, next byte = number of bytes left, then rest of data bytes
PRG file	31, filename bytes . . .	none	as above

terleave sets the increment by which disk sectors are filled during normal DOS I/O. This parameter does not affect the way a file appears to the user because sectors in a given file are linked automatically when the file is accessed.

The *ROM signature analysis* command is basically a ROM self test. If the ROM test fails, the green busy-light will flash four times.

The mode select is used to select either 1541 (single-sided, slow speed) or 1571 (double-sided, slow or fast speed) mode. Either mode can be accessed by any Commodore machine but only the C-128 can use the fast serial data transfer. For example, you can use the 1571 as a double-sided drive on a VIC-20 or C-64. The selected mode remains in effect until either the drive is reset (hard or soft method) or a new mode command is issued. The *side select* command is an interesting feature. With single-sided disks, the data are usually stored on side 0. By selecting side 1 with a single-sided disk (i.e. 1541 mode), you can format it, read it, or write to it, thus creating a pseudo double-sided disk. This is similar to a flippy (the process of turning a disk over to use the second side on a single-sided drive), except that the flip is electronic rather than mechanical. It should be noted that the flip side of a flippy *cannot* be read using this method because the direction of disk rotation changes for a mechanically flipped disk but not for an electronically flipped disk.

The final utility command is the *device number change*. This is identical to the 1541 command and is self explanatory.

Many of the commands will return a value called the 'status byte'. A bit by bit description of this byte is provided in the 1571 disk drive manual. In general, the low order 4 bits represent the status of the disk controller. A decimal value of 0 or 1 for these four bits indicates that everything is fine. Any other value indicates a controller error as listed in the 1571 manual. The high order 4 bits are used for MFM disks. If these bits are set then the disk is MFM format. The bits represent the number of bytes per sector. The status can be checked at any time using the *inquire status* command.

Table 1 is a summary of each of the burst mode commands along with the byte sequence required to access it. All byte values are in decimal. If these commands are sent using a BASIC command string in an **open** or **print#** statement, the sequence should be sent as **"u0" + chr$(a) + chr$(b) + chr$(c)...** where **a**, **b** and **c** are the byte values listed in the table. Similarly, for machine language calls, the listed byte values should be preceded by the values 85, 48 (decimal) or 55, 30 (hex). It should be noted that error checking is not performed on any of the parameters before they are passed to the 1571. This must be done by the calling program before the bytes are sent. Where no burst data input or output are requested, the function can be called entirely from BASIC.

Some of the command bytes have been simplified by selecting the most freqently used options. Other options may be available for some of the commands. A detailed bit by bit description of each command string can be found in the 1571 manual.

The burst transfer protocol is really quite simple. However, it cannot be used with a simple **GET#**, **INPUT#**, or **PRINT#** statement from within BASIC. Data are transferred directly to and from the data registers of the serial bus controller (CIA #1). The process is entirely under user control (i.e. fully manual) and in all cases is handled on a byte-by-byte basis. □

THE COCKROACH TURBO-ROM

A plug-in replacement ROM for the 64, 128 and SX64

- **Fast LOAD, SAVE, VERIFY, FORMAT**
- Innocent SAVE with REPLACE
- Screen dump during execution
- DOS Wedge:

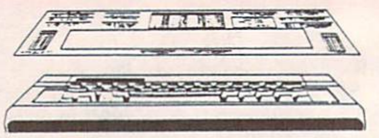
disk status; non-destructive directory with wild cards; scratch, rename, initialize, validate; clear directory and free BAM; change disk device number; @P/@: open 4,4:cmd4/print #4: close4; load with SHIFT/RUN-STOP; cold start, unnew; compatible with VIC switch (LOGO loads in 35 secs); has no tape or RS232 routines but don't worry; you get two ROMS in one - TURBO plus original kernal - toggle between them

$60. CDN. $45 U.S.

Cheque, money order, VISA or MASTERCARD

kobetek systems limited
1007 Commercial Street
New Minas, N.S. B4N 3C4
(902) 678-9800

LEROY'S CHEATSHEET®



NEW!! NEW!!

COMMODORE 128
KEYBOARD OVERLAYS
DIE-CUT — PLASTIC LAMINATED

Forget about constantly referring back to the manual for operating commands. Discover all the "special" commands buried in the manuals. LEROY'S CHEATSHEETS® are sturdy plastic laminated help sheets designed for use with popular software, hardware and languages for the C128 computer. Commands are grouped together for logical use. You learn and use your programs faster and easier.

FREE KEYBOARD EXTENDER
WITH EACH ORDER (A $2.95 VALUE)

only **$7.95** *each*

☐ BASIC 7.0
☐ BLANKS (SET OF 3)
☐ DISK 1571
☐ EASY SCRIPT
☐ FLEET SYSTEM 2&3
☐ PAPER CLIP 128
☐ WORDPRO 128

QTY. ________ X $7.95 = $____.____
SHIPPING $ 1.00 *
6% TAX (PA. ONLY) $____.____
TOTAL $____.____

* U.S. & CANADA $1.00
FOREIGN ORDERS $3.00
CHECK, M.O., MC/VISA
U.S. FUNDS NO C.O.D.

MC/VISA# ____________
EXPIRATION DATE ____________
NAME ____________
ADDRESS ____________
CITY ____________
STATE ________ ZIP ________

VISA
CHEATSHEET PRODUCTS INC.
P.O. Box 111368 Pgh., Pa. 15238
(412) 781-1551
MasterCard

Calendar of TPUG Events

Meeting Places

Brampton Chapter: Brampton Public Library, Four Corners Branch, 65 Queen St., on the second Thursday of the month, at 7:30 pm.

Business Chapter: TPUG Office, 101 Duncan Mill Rd., Suite G-7, Don Mills, on the second Wednesday of the month, at 7:30 pm.

Central Chapter: The Central Chapter will no longer be meeting.

COMAL Chapter: York Public Library, 1745 Eglinton Ave. W. (just east of Dufferin) on the fourth Thursday of the month, at 7:30 pm in the Story Hour Room (adjacent to the auditorium).

Commodore 128 Chapter: York Public Library, 1745 Eglinton Ave. W. (just east of Dufferin), on the first Wednesday of the month, at 7:30 pm in the Story Hour Room.

Commodore 64 Chapter: York Mills CI, 490 York Mills Rd. (east of Bayview) on the last Monday of May, and the second Monday of June, at 7:30 pm in the cafetorium.

Communications Chapter: TPUG Office, 101 Duncan Mill Rd., Suite G-7, Don Mills, on the fourth Wednesday of the month, at 7:30 pm.

Eastside Chapter: Dunbarton High School (go north on Whites Rd. from the traffic lights at Highway 2 and Whites Rd. to next traffic lights; turn left to parking lots) on the first Monday of the month, at 7:30 pm.

Hardware Chapter: TPUG Office, 101 Duncan Mill Rd., Suite G-7, Don Mills, on the second Tuesday of the month, at 7:30 pm.

New Users Chapter: TPUG Office, 101 Duncan Mill Rd., Suite G-7, Don Mills, on the third Tuesday of May, and the third Monday of June, at 7:30 pm.

SuperPET Chapter: York University, Petrie Science Building (check in room 340). Use north door of Petrie to access building. On the third Wednesday of the month, at 7:30 pm.

VIC 20 Chapter: York Public Library, 1745 Eglinton Ave. W. (just east of Dufferin), on the first Tuesday of the month, at 7:30 pm in the auditorium.

Westside Chapter: Clarkson Secondary School, Bromsgrove just east of Winston Churchill Blvd., on the third Thursday of the month, at 7:30 pm.

TPUG makes every effort to ensure that meetings take place when and where scheduled. However, unforeseen problems may occasionally arise that lead to a particular meeting being changed or cancelled. The TPUG meetings line (445-9040) is the best source of fully up-to-date information on meeting times, and should be consulted.

Are you interested in organizing some other interest group in the Greater Toronto area? Please let the club office know, by mail, phone or TPUG bulletin board. □

MAY

MON	TUES	WED	THURS
			1
5 Eastside	6 VIC 20	7 C-128	8 Brampton
12	13 Hardware	14 Business	15 Westside/Amiga
19	20 New Users	21 SuperPET	22 COMAL
26 Commodore 64	27	28 Communications	29

JUNE

MON	TUES	WED	THURS
2 Eastside	3 VIC 20	4 C-128	5
9 Commodore 64	10	11 Business	12 Brampton
16 New Users	17	18 SuperPET	19 Westside/Amiga
23	24	25 Communications	26 COMAL
30			

A Super-OS/9 Bug Report

by Avygdor Moise

Most of the information in this article was derived from a longer list of bugs that was mailed to TPUG by Microware. On the original list, Microware reports problems not only with OS-9 Level I, but also all the known problems with OS-9 Level II for the 6809 micro processor, OS68K (OS-9 for the 68000, 68010 and 68020 processors) and the development package.

In addition, users of Super-OS/9 (OS-9 Level I on the SuperPET) have reported more bugs, some of which exist in the OS-9 operating system itself, and some of which were introduced when TPUG ported OS-9 onto the SuperPET.

An attempt to fix all known bugs has been made, where possible. We hope that our next release of OS-9 (Version 2.0), expected early in the summer, will resolve most of the problems. An announcement will be made in the magazine as soon as Version 2.0 is available. The disk will be available to all Super-OS/9 users on request, at a charge not exceeding twice the cost of a regular library disk.

Some of the bugs in the following list have already been corrected. These are marked with an asterisk, and are accompanied by a short explanation of the cause of the bug, and a description of the fix.

Vendor's Bug Report

The following section lists bugs reported for OS-9 Level I, V1.2 for 6809 operating systems. It is condensed from a list provided by Microware, and dated January 17, 1986.

OS-9 System

Programmer's Manual

• The descriptions of the D and X registers are reversed for the F$AllBit system call.

• The F$AllRAM function has an output: D = Beginning RAM block number. Page 12-3 states that there is no output.

User's Manual

• The documentation for the Proc utility is wrong. Proc does not list age, status or signals.

OS9P1

• **F$Fork call:** When the I$Dup system call (called by F$Fork) fails, it may crash the system, because F$Fork uses the B register as a counter, and that register gets an error code back from I$Dup.

OS9P2

• **F$Unlink call (*):** The Unlink subroutine has a bug that causes I$Detach not to work. This is because the X and Y registers have inverted values (they need to be exchanged). This causes a problem in IOMan. When I$ is passed an invalid pointer, no error is returned.

CMDS

DSave

• The DSave utility does not maintain the attributes or owner number of a directory. This is not a bug, but you should be aware of it.

Link

• Trailing spaces in the parameter list causes an error 235 to be returned. This may be a problem with F$PrsNam. However, this is not a problem with Level I, so it does not directly affect Super-OS/9 users.

Login

• After a user logs in, if a **ctrl-a** is hit immediately, the password is displayed. This is a potential security problem.

Login, TSMon

• The current data directory must contain the **sys** directory in order for **login** to be able to find the **password** file. In order to ensure that the **tsmon** command succeeds, the user must issue a **chd** command to the directory containing **sys/password**.

Shell

• The Shell gets confused when certain characteers are used in command arguments:

format /dx <options> "What Luck!"

format /dx <options> "Ball & Crane"

The "!" and "&" inside the quotes in these commands are interpreted by the Shell as the pipe and the background task characters respectively. This is not really a bug, since the Shell is not designed to recognize double quote (") as string delimiter.

Basic09 (User's Manual)

• The **int** function description is incorrect. The documentation states that the function returns the next smallest whole number, but **int(-2.1)** returns -2 instead of -3.

• The syntax definition for the **renumber** command is incorrect. The ",<incr>" is optional only if the beginning line number is present. This means that the syntax should be:

r[*] [<beg line #> [,<incr>]] <cr>

• The examples for the **print using** statement do not work. In particular:

print using "r5.1", "9999999" produces an error;

print using "e12.3", 1234.567 generates 1.235E+03;

print using "e12.6>", -0.001234 generates ************.

Microware C (Manual)

• In the index, it claims that there is a **toascii** function on page 4-32. This function, however, is not documented in the manual.

• The first paragraph of page 1-2 states that a copy of Kernighan and Ritchie's *The C Programming Language* comes with each copy of the C compiler. This is not true.

Microware C (Compiler)

• The source line **if ((fred & 0xff) = = 0)**, where **fred** is an integer, results in the assembly code:

```
ldd 4,s
clra
lbne 7
```

which, of course, never branches.

TPUG Bug Report

The following section lists bugs reported by Super-OS/9 users. These bugs should be fixed in Version 2.0.

Drivers

ACIA *

• When the receive buffer gets full, Super-OS/9 will hang.

This problem has been resolved by setting the interrupt priority of the ACIA to be lower than that of the console, thus allowing the console to function even when the ACIA is not in use, or when its input buffer overflows, and there is no reader.

In simpler terms, when a character appears in the serial port, an interrupt is triggered. If there is no program reading it (like **XCom9**), or the program accepting the data from the ACIA is too slow to read the data, an alarm clock is permanently set until a process realizes that one or more bytes are waiting to be read from the port. This alarm may consume all of the CPU time if it has too high a priority. By lowering the alarm priority to be the lowest in the system, the interrupt would be serviced if and only if there is a program reading the data and the system console (/term) is not busy.

This has not yet been implemented at the user level. Those of you who have acquired the extended source code package will find a conditional directive in the 'acia.irq' module that allows the user to enable or disable MODEM control on the ACIA port.

• It is not possible to disable MODEM control (CTS/RTS/DCD) in this version. The symptom is that your computer will appear to lock up if your modem is not turned on, or is improperly connected.

• It is not possible (from the driver level) to generate a BREAK signal.

This will be added in release 2.0

CbmDsk *

• A disk drive must be accessed at least once before formatting it or writing to it. This applies to /d0, /d1, /d2, /d3 and /dram.

In the 'init' subroutine, the X and U registers were not used correctly.

• Task switching is not allowed during disk accesses.

This will be taken care of by the use of a disk cache (that we have found to improve the disk access by up 20 times) and by possibly letting the IEEE-488 handlers be interrupt-driven (yes, it *is* possible on the SuperPET).

CbmCon *

• When the Shell is idle, pressing **ctrl-c** or **ctrl-e** will echo "Error 252" and "Error 253" instead of "Error 3" (keyboard abort) and "Error 5" (Quit), respectively.

In the 'read' subroutine, register B was used in the set carry instruction instead of register A. Register B normally holds the error number.

• When input is expected by any utility, pressing the **rubout** (repeat) key will make the cursor vanish.

The terminal driver does not treat the **rubout** characer as a NOP. In 2.0, **rubout** will be displayed as a reverse character (Greek letter *mu*).

• Right after Super-OS/9 boots, two cursors are visible on the screen.

This is a feature, not a bug.

• There is no way, in this version, to generate the special block graphic characters on CbmCon.

By adding a **learn** function to CbmCon, the above problem will be resolved in version 2.

• There is no facility for reading screen status information (like window size, cursor position and character under cursor).

Save Cursor and Select Window functions will be added to Version 2.

* * *

If you have any problems or bugs, or just wish to express an opinion, please write to me in care of TPUG (address is listed in the magazine), and I will be happy to answer. (Better yet, why not attend the SuperPET monthly meetings at York university and get the answers first hand.) If the questions relate to matters that may concern the membership at large, I will reply to them in this magazine. □

Super-OS/9* Is Here

TPUG has implemented the popular 6809 operating system OS-9* on the SuperPET. Super-OS/9 greatly expands the software availability and the hardware capability of the SuperPET while preserving access to the Waterloo languages and programs.

The cost of Super-OS/9 to club members is $210 (Cdn) (plus $10 shipment/handling Ontario residents add 7% PST), which includes the cost of a hardware modification that will not affect the normal operation of your SuperPET, installation instructions and the operating system disks.

To obtain your copy please send your cheque or money order to:

TPUG
101 Duncan Mill Rd., Suite G-7
Don Mills, Ontario
Canada M3B 1Z3

What does Super-OS/9 offer?

- A true disk operating system with features found on UNIX* systems and on the AMIGA.
- Multi-tasking and multi-user capability.
- Hierarchical directory structure with time/date stamps.
- Programmable file access privileges for increased security.

Extensive software is available for OS-9, most of which runs on Super-OS/9.

Super-OS/9 V1.1 includes an assembler, editor, symbolic debugger, communication software and terminal emulation package. Available languages include BASIC09, CIS COBOL, Fortran 77, Pascal, Lisp, C and others. Application software include wordprocessors, spelling checkers, data bases and spread sheet programs.

TPUG has acquired public domain software and will assist users in the conversion of commercial software to Commodore format.

Portability and Expandability

- Super-OS/9 programs will run on all OS-9 based computers (like the CoCo).
- Super-OS/9 will support standard disk drives (IBM format) and the Hi-res graphics board (700 x 300 dots).
- Super-OS/9 software is C compatible with OS-9 68k and AT&T Unix system V.

For further information call TPUG Inc. at (416) 445-4524, ask for Bruce.

NOTE: If you own a 3 board SuperPET and wish to acquire Super-OS/9, please call TPUG before, ordering Super-OS/9, for info about a hardware fix to a design error in your SuperPET computer.

Super-OS/9 is a trade mark of TPUG and Avygdor Moise.
OS-9 is a trademark of Microware and Motorola.
UNIX is a trade mark of Bell Laboratories.

Tele-Communications Package

A complete package for several uses. One disk includes 64 Swifterm, 128 Swifterm and the War Games auto dialer. Three packages all for the price of one!

\$19^{95}

128 80-COLUMN ADAPTER

A dual purpose adapter that gives you 80-columns of monochrome text in the 128 mode. A switch is provided to toggle between 40 and 80 mode. Both sound and video cables are included for a complete hook-up.

\$19^{95}

SPECIALS

Disk notcher **$5.99**
Double your disk capacity with this little tool!
Generic Disks DS/DD . . . . **79¢**
300 Baud Modems with Swifterm and Quantum Link **$39.95**

64-128 CROSS REFERENCE BOOK

This is the first book available of its kind. Complete cross-references used to covert 64 programs over to the 128 computer. Book is formated on the left side with the location and label name, and then on the right with the equivlent 128 location. The entire range of memory is covered starting at 0-Page going thru BASIC and then thru the Operating System. A must for the serious 128 user.

\$12^{95}

MAGNUM LOAD

Programed by Jim Drew

MAGNUM LOAD is a new replacement KERNAL (operating system) ROM chip for your Commodor 65 or 128 computer that will load and verify programs up to 6 times faster than before. The tape routines have been removed from the old chip and in their place have been put a high-speed loader, high-speed verify, and disk driver no-head-rattle routine. Unlike the older cartridge fast loaders, no ports are tied up at the rear of the computer, the screen is not blanked during loading and there is no wear-and-tear on the expansion port. For maximum convenience and performance, the chip is installed directly in the circuit board. Generally a socket has already been provided to make the operation easy, but occasionally some soldering may be required. Now you can give your 1541 disk drive "1571 speed."

Rather than give you more exagerated claims about how many times faster our ROM chip is compared to the slower cartridge versions, a comparison chart is supplied listing MEASURED loading times.

Program	Star-Dos	Reg. Load	Mach 5	Fast Load	MAGNUM LOAD
Pitstop II	?	144 sec.	43 sec.	41 sec.	**31 sec.**
Music Shop	?	105 sec.	105 sec.*	105*	**21 sec.**
Hitchhiker's Guide to the Galaxy	?	70 sec.	70 sec.*	N.G.**	**68 sec.**
On-field Football	?	149 sec.	66 sec.	63 sec.	**56 sec.**
EASY FINANCE I	?	58 sec.	13 sec.	13 sec.	**11 sec.**

\$24^{95}

For 1541 or MSD Version

*Will not fast load - defaulted back to regular load

**Failed to load at all

A Disk - A 2nd drive Imulator for the Amega **$49.95**
A-Filer - A complete File PRG for the Amega **$49.95**
A Report - A complete Report Writer for the Wmega **$49.95**
A Term - Excellent, easy to read terminal package for the Amega **$49.95**

64 BBS

Full performance boards with lots of extra's not found on others. Both boards have message rooms that can be setup for a variety of different uses. Each room can have it's own access level assigned for maximum control.

- 7 read and write rooms with up to 8 secur ity levels
- Public message base and sysop's mail box
- Secret highest level for complete access of all functions.
- Auto cycling of E-mail.
- Remote access for system operator. Boards activity can be sent to printer.
- 300/1200 baud selectable
- Up and down file transfers with a choice of access levels
- New punter and X-modem protocol supported.

\$39^{95}

D-CODER

- Translates any machine language program into easy-to-read English descriptions with complete explanations of each command.
- Makes complete notations of all important memory locations accessed by the program (SID, VIC, MOS, KERNAL, etc.)
- Gives you three ways of accessing programs:
  1. **Will read and list programs from DISK**
  2. **Will read and list programs from MEMORY**
  3. **Direct user input (from magazines, etc.)**
- Can be used to locate and examine any machine language program's protection routines!
- Can be used to easily break apart machine language programs for study and examination!
- Printer option for complete hard copy listings!

You no longer need to be an EGGHEAD to read Machine Language.

\$19^{95}

128 BBS

This is the first complete 128 bulletin board of it's kind. All the features of the 64 board and more. The board takes advantage of the extra memory by providing a complete seperate sub-board. User can have both boards available at the same time. Imagine running an open board for everyone a secret closed board for the more serious users. Look at these features

- Print out boards activity to include what files accessed and by who.
- Faster performance and more storage using the 1571 drives.
- Supports the use of over 500 users!
- Both boards will support and work on any type of disk drive including the big 1 meg. SFD's.

\$39^{95}

Both modems work on the 1650, 1670 and their compatibles.

N-CODER

THE PERFECT COMPANION PROGRAM TO D-CODER!

Allows you to easily make changes in machine language programs . . . right on the disk!

- Rewrite ability allows code to be altered and then rewritten directly to the disk!
- Features sector-by-sector scrolling assembly language display of machine language programs!
- Notation of ASCII text equivalents for easy spotting of embedded text strings!
- Handy reference display of all assembly language commands and their ML numerical equivalents!
- Byte splitter for easy splitting of decimal addresses into low byte-high byte format!

\$19^{95}

DIR+
A DISK LIBRARY CATALOGING PROGRAM

Having problems finding that lost disk? Here's a program that allows you to alphabetize and print out your program names in a versatile format. A scrolling arrow allows you to edit out meaningless names and keep what you want. Know what you got and where its at!

- 60 to 600 titles (1 to 5 COLUMNS) can be printed per page by using optional condensed printing.
- Print disk labels with your disk names and ID's.
- Can support up to 1100 program titles and can alphabetize in less than 15 seconds!
- Save the master list to disk for use with a word processor or a data base.

ONLY \$19^{95}

Dealer and Distributor Inquiries Invited.

Enclose Cashiers Check, Money Order or Personal Check. Allow 14 days for delivery. 2 to 7 days for phone orders. Canada orders must be in U.S. Dollars. VISA — MASTER CARD — C.O.D.

Programs for C-64/128 — \$3.00 S & H on all orders

Software Submissions Invited

VISA

MasterCard

MegaSoft Limited

MegaSoft Canada LTD.
P.O. Box 10
Parry Sound, Ontario, Canada
Phone 705-746-8081
In USA call 800-541-1541
Telex 4949909/MEGAS

Top Secret Stuff I & Top Secret Stuff II

Programed by Jim Drew

Are both collections of 20 programs per diskette (that works out to about 50¢ per program!) that help you explore and enhance your Comodore 64 and/or 128 and 1541 disk drive. Now you can unlock many secrets formerly known only to top machine language programmers by using these sophisticated "tools." If you have ever been curious about the inner workings of your computer system, now is your chance to dig in and find answers with the help of these programs. These collections of progams have gotten rave reviews from actual users, and we are sure that you, too, will be pleased.

Both for only $19^{95}

TOOLBOX 64 AND NEW TOOLBOX 128

Side 1 contains over 100 routines, some of them are for protection, smooth scrolling, modem routines and sound and color. Also a bootmaker, paddle and joystick test; read terminal, and auto dial and auto answer. Documented routines allow you to use them to build your own programs or use alone. This disk has a lot of tricks that are used in commercial software.

Side 2 contains several of the same routines for the 128 system but also a T&S editor for going out to track 66, along with a screen dump for the 80 colum mode and lots, lots more.

Side 1 C-64 **Side 2 128** All for $19^{95}

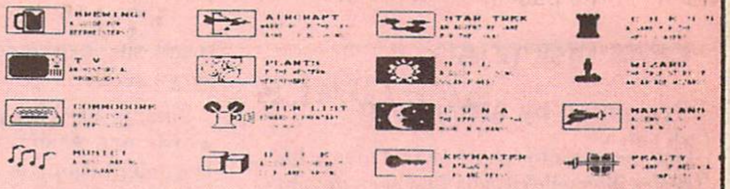
Turtle Graphics II
Bravo
Maze Master
Lazer Zone
Turtle Toyland Jr.
Gridrunner II
Pitfall
Minnesota Fats
Frogger
Q-Bert
Popeye
Quality Joy Stick
Rootin Tootin

While Supplies Last!

Your Choice 9^{95} Each

GRAPHIC LABEL MAKER

Give your labels the professional touch. With Hi Rez Graphics make your own design or use one our 60 premade labels with easy to use on screen editor. You can insert up to three lines of text then choose the picture you want to put on the left hand side of the label. Then you can print out as many labels as you want. This has got to be the neatest label program out there.



*Now works with Print Shop

$24^{95}

PHOTOCOPY

PHOTOCOPY is a graphics integration program which lets you create NEWSROOM photos and banners from other graphics programs. Files also can be converted from NEWSROOM to PRINTSHOP. Photocopy is an idea enhancement to NEWSROOM giving you the versatility to increase your graphics library.

- Photocopy will convert NEWSROOM photos to PRINTSHOP
- Phtocopy works with a variety of files "DOODLE, FLEXIDRAW, PRINTSHOP, SCREEN MAGIC, GRAPHICS BASIC, COMPUTER-EYES".
- Photocopy will convert a NEWSROOM photo or banner to a PRINTSHOP graphic.
- Photocopy works with the 1541, 1571 and the Indus drives.

$19^{95}

Newsroom, Doodle, Printship, Flexidraw, and Graphics Basic are trademarks of Springboard Software, City Software, Broderbund Software, Inkwell Systems, and Hesware respectively.

We'll Send You Money!

That's right — TPUG Inc. will send you — with NO obligation (on our part) — hard cash that you can spend as you please. You don't even have to be a TPUG member! Hard to believe? It's true!

Here's how to take advantage of this special offer: Just send us a self-addressed stamped envelope, along with an original unpublished article suitable for TPUG Magazine. The topic doesn't matter — just make sure that it relates to Commodore computers, and that it's brilliant. We do the rest.

Your article should be on a Commodore disk, preferably with accompanying hard copy. Sequential PETASCII files are best, with no embedded word processor commands, but other formats are acceptable. Just be sure to state both disk format and word processor used on the label of the disk, along with your name and the filename of the article. We'll get back to you as soon as possible, usually within three weeks of receiving your submission.

By the way, if you have an idea for an article, but you're not sure whether it's suitable for TPUG Magazine, check it out with us first. Just call (416) 445-4524 (9 to 5 EST weekdays), or drop us a line at: Submissions Department, TPUG Magazine, 101 Duncan Mill Road, Suite G-7, Don Mills, Ontario, Canada M3B 1Z3.

And remember, if we print your article — we'll send you money!

"Cecil's 6502/6510 Conditional Macro Assembler" and "Assembly Language Source Code Editor" for Commodore 64® or Commodore 128® or Commodore SX64®

A package of programming tools on one 5¼ **DISK** created for every assembly language programmer.

Fully capable for the professional: "'Cecil's Assembler' always EVALUATES THE WHOLE EXPRESSION, THEN RETURNS THE CORRECT HIGH/LOW byte".

Cecil's Editor provides easy access to extensive on-line help screens for a variety of information about this assembler package. On-line help describes the assembler syntax, pseudo-ops and macro instructions, and the editor commands.

- *Features 10 built-in macro instructions to complete the branch or jump on condition instruction sets.*
- *Labels may be up to 13 characters in length. All 13 characters are considered significant.*
- *Includes full-screen editor with enhanced DOS commands.*
- *Nests macros to 32 deep.*
- *Includes 10 pseudo-ops to control the assembly process.*
- *Stops on error and/or identifies error type and location.*
- *Supports most printers and interfaces and allows multiple disk drives.*
- *Prints its own instruction manual.*

ONLY $49.95 Cdn. or $34.95 U.S. by Check, Money Order, Visa or MasterCard to:

NORTH OHIO FIRMWARE
Box 2661
North Canton, Ohio 44720-0661

( ) English	( ) C64*
( ) German	( ) C128* in C64* Mode
( ) French	( ) SX64
( ) Information $1.71 Cdn.	$1.00 U.S.

Or Call 216/497-0736 from 9:00 a.m. to 11:00 p.m. EST.

Commodore 64, Commodore 128* and Commodore SX64* are registered trademarks of Commodore Business Machines.*

Library Additions

CP/M Disk (Z)AC

Presented by Adam Herst

Only one disk to report on this month but, given the panicky calls I've been getting, a little information concerning the use of CP/M disks is in order.

CP/M disks *must* be used in CP/M mode. While a directory of sorts can be pulled from a CP/M disk in 128 mode, it indicates that there are no files on the disk with 0 blocks free. If you get a disk you think has no files on it, try accessing it in CP/M mode.

In CP/M mode, the commands with which you are familiar from BASIC do not work. You cannot use **load, save, list** or **run** with CP/M programs. To pull a directory from a CP/M disk you must use the **dir** command (try the **f3** key). To run a CP/M program, you merely type its name at the prompt. If this file is a **.com** file it will be loaded and automatically begin execution.

Since there is no **list** command, the format of the standard TPUG **list.me** file has been slightly modified. These files are now called **type.me** files, because CP/M *does* possess a built-in **type** command. To view the **type.me** file enter type **???/type.me** at the prompt.

On this month's disk are a number of utility programs donated by TPUG member Ora Flaningam. **D, SD** and **ZX** are all directory programs that are smaller and faster than **DIR. ZX** comes in a **.lbr** file, and can be extracted using the library utilities on this disk. **DU** is a disk track and sector editor. Designed for a Kaypro disk, it should work on C-128 disks. Documentation is in **DU-V83.DOC. SZP** is another disk editor.

LDIR, LRUN, LU, LUX, LUXDIR, LUXTYP are all utilities for manipulating library files. **LU300.DOC** should tell you everything you want to know about library files. **SQ** and **USQ** are recent versions of the **Squeeze** and **Unsqueeze** utilities with copious documentation. Starting in the near future, programs on TPUG CP/M disks will be distributed as squeezed library files to save on limited disk space.

NSWP is a more recent version of the **SWP** file copy utility. Documentation provided in **NSWP.DOC. MFT** is a high-speed, single-drive file copier. This one is very useful!

Finally, **VDO** is a good, **Wordstar**-like word processor and text editor. Documentation is provided through online help menus and many of the **Wordstar** commands are honoured.

Next month should see the release of a number of public domain languages for CP/M. Remember to send us any public domain programs you may have. In return we will send you a disk of your choice for each one we receive and use.

```
zac/type me  : du       com
du-v83   doc : ldir     com
library  lbr : lrun     com
lu       com : lu300    doc
lux      com : luxdir   com
luxtyp   com : mft      com
mft      doc : nswp     com
nswp142  doc : sq       com
sq/usq   doc : szp      com
szp      doc : usq      com
usq      doc : vdo      com
vdo      doc : zx       lbr
```

SuperPET Disk (S)TZ

Presented by Bill Dutfield

Finally, to complete the additions to the SuperPET Library, a SuperPET disk was released in December. This disk, contributed by Alain Proux, contains a tutorial on writing math functions in 6809 assembler. There are nine modules, supported by three demonstration programs, covering: accumulators and registers, Integers (2 modules), floating point routines, first steps in FP arithmetic, output formatting, advanced inputting, input and output operations and comments on demo programs.

```
0     superpet disk (s)tz
3     "describe.dec/85"      seq
1     "-maths with the-"     seq
1     "---assembler----"     seq
1     "--------1--------"    seq
1     "--editor files--"     seq
1     "--------a--------"    seq
1     "--introduction--"     seq
17    "mat0:e"               seq
1     "---tutorials----"     seq
35    "mat1:e"               seq
37    "mat2:e"               seq
18    "mat3:e"               seq
44    "mat4:e"               seq
26    "mat5:e"               seq
52    "mat6:e"               seq
56    "mat7:e"               seq
38    "mat8:e"               seq
36    "mat9:e"               seq
1     "--------b--------"    seq
1     "--source code---"     seq
1     "----integers----"     seq
14    "integers.asm"         seq
1     "integers.cmd"         seq
1     "----storage-----"     seq
20    "storage.asm"          seq
1     "storage.cmd"          seq
1     "------demo------"     seq
14    "main.asm"             seq
14    "routines.asm"         seq
40    "input.asm"            seq
38    "display.asm"          seq
14    "formating.asm"        seq
25    "save.asm"             seq
10    "variables.asm"        seq
1     "demo.cmd"             seq
1     "--------2--------"    seq
1     "-main menu prg--"     seq
2     "integers:men"         prg
3     "storage:men"          prg
20    "demo:men"             prg
1     "--------3--------"    seq
1     "sample data file"     seq
1     "sample:dd"            seq
```

C-128 DISK (Y)AB

Presented by James Kokkinen

If you have developed C-128 programs for the public domain that you feel are worth sharing with TPUG's membership, we are anxious to review them for possible inclusion in future monthly or specialty disks. Programs that include documentation on disk are preferred.

This is the second disk entered into the TPUG C-128 library for operation in C-128 Mode. It contains some examples of the types of programs that can be developed on this computer in both BASIC 2.0 and BASIC 7.0. This disk contains four games, all educational, a collection of clever sayings and a wealth of well-documented utilities for reading, formatting, programming, and booting your programs in C-128 mode for both BASIC and CP/M formats as applicable.

It is always advisable to back up the programs you use from these disks onto disks formatted on your own equipment prior to running them.

We open with **Number Invaders**, an educational math game for players of all ages. You set the level of difficulty, give your name, and the game keeps track for you, providing rewards for exceeding your previous score. **Scores-Num.I** and **Racecar4.spr** are sub-programs of this game.

Factor Race.y is a more advanced math game adapted from the C-64 library to run on both 40 and 80 column screens. **Geography.y**, and **Definition.y** are both word games including documentation to provide hours of educational fun. **Proverbial.y** is a collection of short sayings, quotations, and anecdotes. It is best viewed on the 40 column screen.

Moving on to utilities, we have **Filecon.y** and its partner **Filecon.ml**, which can be used to convert BASIC text files to CP/M text files and vice versa. **Filecon.ins** is a sequential documentation file for **Filecon.y**. **IBM-Filecon-lst** is a short program you can type to an IBM formatted diskette to perform similar conversions on IBM PCs. **IBM-Filecon-ins** tells you how. **Autobootmaker2.y** is what its title implies, with documentation as to its use in **Autoboot.ins**.

Hi-Res Text.y allows bit mapping of oversize text on graphics screens. This is for use within other programs, and may require operator adjustments in order to achieve the desired results. **Hi-Res.ins** explains.

Mergekey setup.y provides a one-key command to append program files — handy for writing long, multipart programs. Documentation is provided in **Merge.ins**. **Key2.ed.y** is an edit program described in **Keyed.ins**. **MFMformat2.y** allows the formatting of disks in several different CP/M MFM formats while in C-128 mode. **MFMdisk.ins** is the documentation describing this one. **CPM Block.y** allows the viewing of the block allocation of CP/M disks while in C-128 mode. Its usefulness and limitations are described in **Block.ins**.

All in all, this is a well-rounded package to enhance any collection.

```
      c-128 disk (y)ab
8     "list-me(y)ab"         prg
1     ".....games....."      prg
40    "number invaders"      prg
1     "scores-num.i."        seq
3     "racecar4.spr"         prg
21    "factor race.y"        prg
25    "geography.y"          prg
89    "definition.y"         prg
52    "proverbial.y"         prg
1     "...utilities..."      prg
15    "filecon.y"            prg
2     "Filecon.ml"           prg
17    "filecon.ins"          seq
20    "ibm-filecon-lst"      seq
10    "ibm-filecon-ins"      seq
4     "autobootmaker2.y"     prg
5     "autoboot.ins"         seq
6     "hi-res text.y"        prg
16    "hi-res.ins"           seq
2     "mergekey setup.y"     prg
3     "merge.ins"            seq
7     "key2.ed.y"            prg
2     "keyed.ins"            seq
6     "mfmformat2.y "        prg
6     "mfmdisk.ins"          seq
2     "cpm block.y"          prg
5     "block.ins"            seq
```

VIC 20 Disk (V)TR

Prepared by Richard Best

February's addition to the VIC Library contains a concentration of educational programs, but everyone should be able to find something useful or interesting here. Many of the programs do not require any expansion.

The disk/tape starts out with an interesting game called **rescue**, wherein you must rescue little men who have been trapped on ledges inside a cave. The cave is patrolled by deadly flying dragons. The one other game here is **football** for an expanded VIC. The screen display is a bit sparse, but the flavour of the game is maintained.

In the utility department we have **pause**, which turns the left shift keys into pause keys. Use this one to stop and restart program listings when debugging. **easter** will calculate the date of Easter Sunday for just about any year.

comic file is a nicely written file manager for tape users. It helps you to create and maintain a data base of your comic book collection. **file mast 2** is an upgrade of **file master**", which appeared on (V)TL. This 'scrunched' version allows you to create a larger file when using only 8K of expansion. You will need a disk drive.

We have two SuperExpander demos that illustrate animation with perspective. Each moves a flying saucer from one corner of the screen up to the front. **ufo-1**

The Walker

1) Insert your COMAL disk in drive*.
2) Type LOAD "C64 COMAL*",8
3) Type RUN (starts COMAL)
4) Type AUTO
(COMAL provides the line numbers)
5) Enter the program lines shown below
(COMAL indents lines for you)
6) Hit RETURN key twice when done
7) Type RUN
Watch an animated sprite hobble across the screen. Change the (99) in line 450 for really fast walking

```
0010 setup
0020 repeat
0030   walking
0040 until key$="q" //Q to Quit
0050 //
0060 proc setup
0070   blue:=14; pink:=10
0080   white:=1; black:=0
0090   define'images
0100   repeat
0110     input "speed (1-10): ": speed
0120   until speed>=1 and speed<=10
0130   background black
0140   setgraphic 0
0150   spriteback blue,pink
0160   spritecolor 1,white
0170   spritesize 1,false,false
0180   plottext 1,1,"press q to quit"
0190 endproc setup
0200 //
0210 proc define'images closed
0220   dim shape$ of 64, c$ of 1
0230   shape$(1:64):=""
0240   shape$(64):=chr$(1)//multicolor
0250   c$:=chr$(0)
0260   for x=22 to 63 do shape$(x):=c$
0270   c$:=chr$(170)
0280   for x=1 to 21 do shape$(x):=c$
0290   define 0,shape$
0300   c$:=chr$(20)
0310   for x=22 to 42 do shape$(x):=c$
0320   define 1,shape$
0330   define 3,shape$
0340   c$:=chr$(60)
0350   for x=43 to 63 do shape$(x):=c$
0360   define 2,shape$
0370 endproc define'images
0380 //
0390 proc walking
0400   for walk:=1 to 319 div speed do
0410     x:=walk*speed
0420     y:=100+walk mod 4
0430     spritepos 1,x,y
0440     identify 1,walk mod 4
0450     pause(99)
0460   endfor walk
0470 endproc walking
0480 //
0490 proc pause(delay) closed
0500   for wait:=1 to delay do null
0510 endproc pause
```

*** If you don't have COMAL yet, order a Programmer's Paradise Package-$19.95. It includes the complete COMAL system plus over 400 pages of information. Add $5 more to get our 20 interactive lesson Tutorial Disk. Add $2 shipping. Visa/MC or US funds check accepted. Send to:**

COMAL Users Group USA
6041 Monona Drive, Room 109
Madison, WI 53716
phone 608-222-4432

moves the object by redrawing each line in turn, and **ufo-2** redraws the entire ship. These were written by a physics professor and contain lots of comments and some integral math.

To keep those old gray cells working, we have a nice collection of quizes and tutorials. **brain teasers** is a mini IQ test, guaranteed to tax your problem-solving abilities. If you've been having trouble sorting out numbering systems, **place value** will help by drilling you on the decimal system. A very nice, graphic lesson in fractions is presented in **pie chart**. You will need a SuperExpander, an 8K expander and a 'mother board' to run this one.

Youngsters who are just learning multiplication and division should find **2-Digit Mult** and **2-Digit Div** very helpful. Each drills the student by printing problems on the screen. The problems are printed as they would appear on paper and you solve them just as you would if working on paper. This is especially effective in **2-Digit Div**, which rejects incorrect numbers and provides helpful prompts.

```
      vic disk (v)tr
9     "list-me (v)t-r/1" prg
4     "list-me (v)t-r/2" prg
2     "rescue boot.v"    prg
10    "rescue.v5k"       prg
2     "pause.v"          prg
14    "brain teasers.v"  prg
9     "place value.v"    prg
7     "comic file.v"     prg
12    "2-digit mult.v"   prg
13    "2-digit div.v"    prg
5     "easter.v"         prg
5     "ufo-1.vsx"        prg
5     "ufo-2.vsx"        prg
20    "pie chart.vsx+8"  prg
18    "football.v12k"    prg
22    "file mast 2.v12k" prg
```

Super-OS/9 Disks

Presented by Bill Dutfield

As of the middle of January, the following OS-9 disks have been released to the Public Domain. All the programs have been obtained from the OS-9 User Group. The load modules are to be found in the **CMDS** subdirectory, descriptions of the programs are in the **HELP** subdirectory and the source code is in the **SRC** subdirectory.

Utilities Disk S1

The contents of this disk include: **Tree**, a module listing all the files in a Super-OS/9 Relative file, of which there is usually one per disk for the distribution library. **Ap** is an append utility to append a file or to combine several files into one. **Attr chg** changes the attributes of a file. **BootSplit** splits merged object files (such as **OS9Boot**) into separate modules. **Ddir** lists active system devices, their path descriptor address, physical address, system buffer, device driver and file manager. **Dlist** dumps disk sectors.

Dates keeps track of dates, appointments, birthdays and warns when they are coming up. **DocGen2** is a submission form for use when submitting Super-OS/9 Software. **Hcopy** hierarchically copies one directory to another. **Help** displays a help file. **Hdir** is another multilevel hierarchical directory. **Install** adds a file to the system for inclusion at boot-up. **ListN** lists a text file. **Lload** uploads a text file, one line at a time, to a full-duplex bulletin board system. **Make** assists in the control of relinking multimodule files when one module is changed. **ModList** lists the files in a multimodule file. **QDir** outputs a vertical module listing. **RMLocate** calculates an RMS record number for a given key field. **RMNew** is used to generate a blank data file for RMS. **ReHook** moves a file from one directory to another on the same device. **Remote** links the user to a remote terminal path for communication. **Replace** replaces strings in a text file. **Strip** is used to strip, add or process any combination of: CRs, LFs or other control characters. **Wcl** counts the characters, words, and lines in a file using 32 bit numbers.

Adventure Disks S2 and S3

This game (the usual search for treasure in the underground caves of the author's imagination) is released on two disks. The first, S2, contains the executable module and the associated data files, while the second, S3, contains the source code.

Utilities Disk S4

The contents of this disk include three SuperPET programs plus a further assortment of Super-OS/9 files.

The three SuperPET programs are **Test.main**, **Test.banks** and **Test.os9**. These three programs provide a complete set of tests to test main memory and the banked memory on the SuperPET.

The OS-9 programs are: **Bincom** which compares two files on a byte-by-byte basis. **Diskid** rewrites the diskname and date on LSNO. **Dlist** dumps disk sectors. **Grep** is a text search program. **Hcopy** hierarchically copies one directory to another. **Hdel** is a hierarchical delete function. **Kalah** is a game based on ancient Bedouin Arab game. **Mount** is a command to change allocation of Super-OS/9 files where there are two or more on a device (for 8050 or 8250). **Print.b** is a file printing utility. **Replace** replaces strings in a text file. **Texcom** compares two files a line at a time. **Today** lists date and time in a legible format.

Math Library Disk S5

This disk contains a series of math routines for use with the MicroWare C Compiler. They are written in 6809 assembler language for speed, and are computed in double precision.

Xlisp Disk S6

Xlisp is a variation and subset of the Lisp language, used in artificial intelligence applications.

Kermit Disk S7

Kermit is a communications protocol for the transmission of files between computers. This disk contains a version of **Kermit** for use under Super-OS/9. Provided is the source code written in C along with an executable load module and a help file. Kermit works well in conjunction with **xcom9**, provided on the Super-OS/9 distribution disk.

C Language Disk S8

This disk contains: **Delw** to delete a list of files using wild cards. **Pf**, a simple format utility to output text such as source code to the printer. **Ppc** is another program, functionally richer, to format output for printing. **Print.c** is yet another output formatter, with even more features.

The following five programs work with various 6809 assembler code found in the various implementations of 6809 systems. **S1flex** is a program to build a *flex* binary file from an S1 input file. **S1intel** converts S1 format text file to Intel format. **S1load** builds a binary file from an S1 input file. **S1unflex** converts a **flex** binary file to S1 format. **S1unload** is a program to convert straight binary file to S1 format. **Showc** is a utility to make unprintable characters in a text file visible. **Tcmp** compares two text files, printing differences, and attempts to synchronize after finding differences. **Xc** is a cross reference and printing utility for C programs (it doesn't work on its own, but does work on other, smaller files.) **Xc.com** is a batch file for compiling **xc** using /DRAM and /D1. □

Software order form

NAME ______________________

STREET ADDRESS ______________________

CITY/TOWN/P.O. ______________________

PROV/STATE ____________ POSTAL/ZIP CODE ____________

TELEPHONE ____________ MEMBERSHIP NO. ____________

TORONTO PET USERS GROUP, 101 Duncan Mill Road, Suite G7, Don Mills, Ontario M3B 1Z3 416-445-4524

disks

To order club disks by mail, send $10.00 for each 4040/2031/1540/1541 disk (4040 format), discount price 5-10 $9.00 each, 11 or more $8.00 each; and $12.00 for each 8050/8250 disk (8050 format). We do honour purchase orders from school boards.

These disks are for use with a ____________ computer and a ____________ disk drive.

Please send me the following:

3 Letter/No. Code	Description	4040 or 8050 Format	Price

Total $ ______.00

tapes

To order VIC 20 or Commodore 64 library tapes, send $6.00 for each tape.
To order PET/CBM or Commodore Educational Software, send $10.00 for each tape.

These tapes are for use with a ____________ computer and a datasette.

If for a PET computer, what model - ____________ - BASIC - 1.0( ); 2.0( ); 4.0( )?

3 Letter/No.	Code	Description	Price

Total $ ______.00

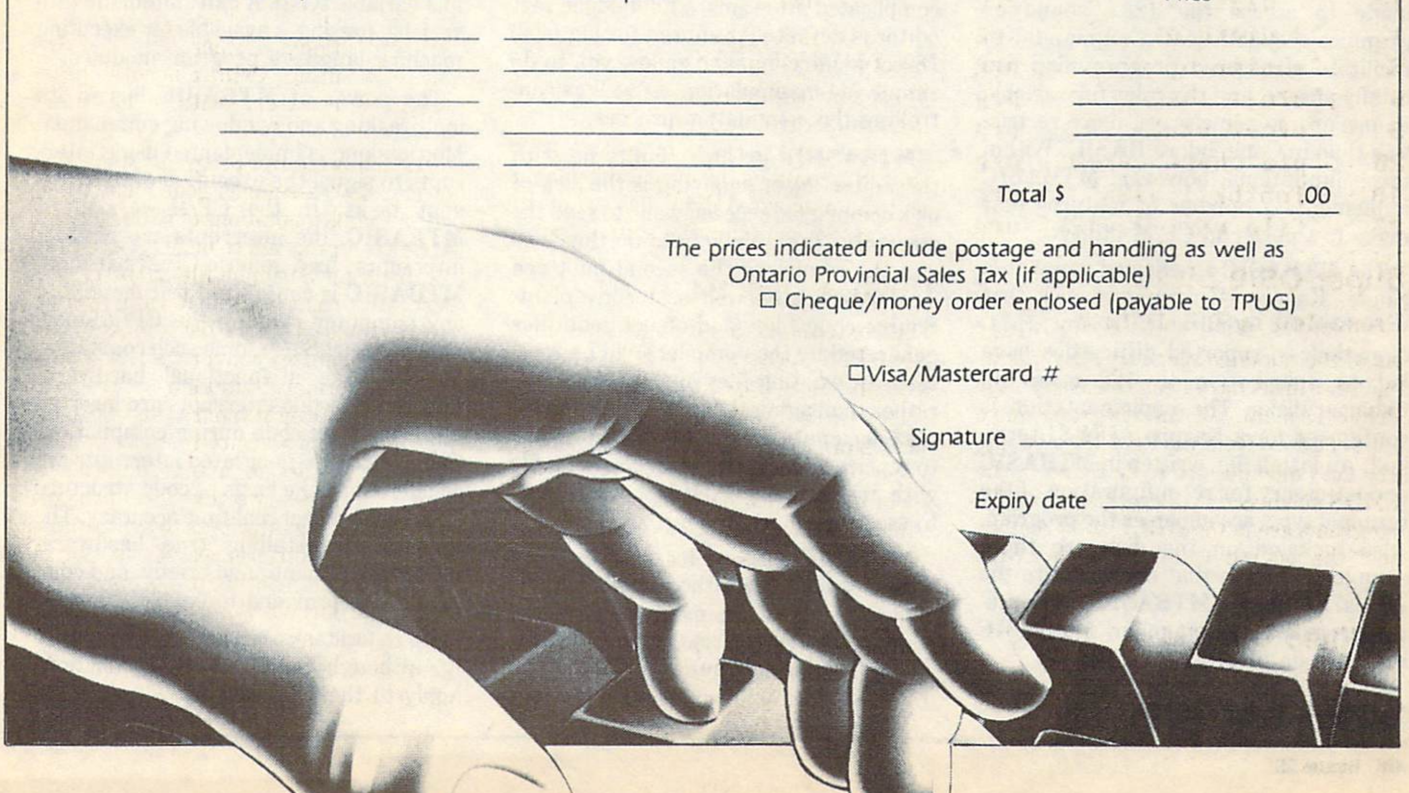
The prices indicated include postage and handling as well as Ontario Provincial Sales Tax (if applicable)

☐ Cheque/money order enclosed (payable to TPUG)

☐Visa/Mastercard # ____________

Signature ____________

Expiry date ____________



Reviews

MTBASIC
from Softaid
Multitasking BASIC Compiler for CP/M on the C-128

Review by Adam Herst

In an issue devoted to alternative languages for Commodore computers, a review of a third-party BASIC seems slightly misguided. If there is one language with which all Commodore users are familiar, it's BASIC, in one of its various incarnations from 2.0 to 7.0. These BASICs are built into the computers themselves and become available on power-up, providing an easy, effective interface to the system hardware. The C-128 in CP/M mode lacks this amenity, BASIC being as foreign to CP/M as Pascal, C or Lisp.

Any implementation of BASIC in CP/M mode will be compared to the powerful BASIC 7.0 available in 128 mode. **MTBASIC** can only be described as sparse when this comparison is made. As a generic CP/M compiler, no effort is made to access the 128's sound or graphics capabilities. Commands to facilitate structured programming are totally absent, and the rules for variable names are, in some ways, more restrictive than in Commodore BASIC. Within these limitations, however, **MTBASIC** implements a number of features that make it a powerful CP/M utility.

The **MTBASIC** I reviewed came on a single, Kaypro IV formatted disk that was handled admirably by my 1571. Nonetheless, reported difficulties have led to distribution to 128 users on Osborne disks. The implementation is configured for a Kaypro ADM-31 terminal. An install file, written in **MTBASIC** and necessary for reconfiguration of the terminal type, accompanies the program. Also included on the disk are three sample programs that demonstrate the special features of **MTBASIC**, and a utility program to renumber **MTBASIC** source code files.

The manual accompanying the disk is comprehensive and easily understood, if poorly organized – installation instructions should not come 30 pages in. A portion of the text concerns the **MTBASIC** implementation on MS-DOS machines and is irrelevant to the CP/M user. The manual is divided into two parts:

• Using **MTBASIC**: this section gives instructions on installing **MTBASIC**; examples of programming the special features of **MTBASIC** like windowing and multitasking; handling disk files and directing I/O channels; and speeding up programs. All of the topics are accompanied by simple programs illustrating the concepts.

• User Reference: This section details the syntax of **MTBASIC** and covers the direct (immediate mode) commands. System details and operating system interfacing are covered.

MTBASIC is invoked by typing **mtbasic** at the CP/M system prompt. When active, **MTBASIC** displays '**>**' as its prompt. While this prompt is displayed, crude line editing commands are in effect, allowing you to type in either program lines (preceded by line numbers) or direct commands. This editor provides seemingly full syntax checking, and will not accept a syntactically incorrect program line. While suitable for short, uncomplicated programs, a full-fledged text editor is definitely required for big jobs! Direct mode commands allow you to do simple file manipulation, as well as controlling the compilation process.

MTBASIC's greatest faults lie with this editor. Most annoying is the lack of disk commands, especially one to read the directory. The only way to do this is to exit the compiler. The second fault can be dangerous as well as inconvenient. Source code files (and object code files generated by the compiler) will be written over existing files of the same name rather than aging the files with a **.bak** or **.$$$** designation. This makes it possible to replace your only correct copy of a file with an incorrect copy! Not a nice way to start your day.

MTBASIC is a compiler, but a compiler with a twist. Once the source code is loaded or typed into memory, it can be compiled to memory and immediately executed by issuing a **run** command. With its quick compile times (see below), **MTBASIC** can act as a pseudo-interpreter. As well, programs can be compiled to memory and not executed, or compiled to disk. Errors during compilation leave the offending line in memory to be corrected. Programs compiled to disk exist as **.com** files, and can be executed independently of **MTBASIC**.

Two aspects of a compiler determine its usefulness: a full implementation of the target language, and good compilation specifications. **MTBASIC** implements a decidely eccentric version of BASIC. No attempt is made to adhere to the Microsoft pseudo-standard. While the syntax is fairly standard, incompatibility is ensured by **MTBASIC**'s requirement that all variables and arrays be declared as **real**, **integer** or **string**. Consequently, foreign BASIC source code requires massive translation before it can be compiled. Also, **MTBASIC** does not recognize differences in variable names across modes. You could not declare both **integer a** and **string a** in the same program.

In extent, the vocabulary of **MTBASIC** is most similar to BASIC 2.0. It surpasses BASIC 2.0, however, with its extensive set of disk and I/O commands, the variety of its predefined functions and its extremely versatile user defined functions (multiple lines, multiple argument, multiple variable type). A **call** statement with variable passing is available for executing machine language program modules.

The power of **MTBASIC** lies in its multitasking and windowing commands. Multitasking is implemented using interrupts to signal the execution of independent tasks. In the CP/M version of **MTBASIC**, the interrupts are pseudo-interrupts. As a generic CP/M package, **MTBASIC** is configured to function on any computer running the CP/M operating system. Since some such computers do not have a functional hardware-interrupt, pseudo-interrupts are inserted into the object code during compilation. These software-generated interrupts are calculated on the basis of code structure and do not reflect real-time accuracy. The process of installing true hardware-interrupts is mentioned briefly, and commands are provided to enable them.

Up to ten tasks can be run concurrently, although fairly severe restrictions apply to their interaction. All variables are globally defined and can be altered by any of the executing tasks. I/O operations must be designed so that no conflict arises between tasks for access to a given device. Powerful windowing commands are implemented which, combined with **MTBASIC**'s multitasking abilities, should allow the creation of sophisticated programs.

The most critical aspect of any compiler is its compilation specifications. Slow compilation times, large object code and slow execution times can make even the most full-featured compiler useless in program development. **MTBASIC** contains a number of commands to control the compilation process. Compilation can be directed to memory or disk, with or without run-time error checking, and with or without the run-time modules required for multitasking. The first command affects the speed of the compilation process, while the latter commands affect the size and speed of the resultant object code.

To benchmark the various compilation times, I used the source code for the install file. This file, which occupied 6K on a CP/M directory, compiled to memory in less than two seconds. Compilation to disk required 81 seconds. To assess execution time, I ran the Sieve of Eratosthenes program that *BYTE Magazine* uses for a standard benchmark. With error checking and multitasking off, the MTBASIC version ran in 155 seconds; it took 230 seconds with error checking and multitasking on. The BASIC 7.0 version ran in 436 seconds in fast mode.

All in all, **MTBASIC** is an excellent package for a number of reasons. As an interface to the CP/M operating system it is both powerful and familiar to the Commodore BASIC programmer. Its ability to multitask provides a unique environment for the development of programs. These should be extr:mely portable across CP/M computers (if the object code doesn't work just install your version of **MTBASIC** on the target computer and recompile the source code) and even to MS-DOS computers (compile the source code with the MS-DOS version of **MTBASIC**). The speed of compilation to memory and the ability to execute this code without leaving the compiler ease program development. Finally, **MTBASIC**'s low price ($39 US) and lack of copy protection make the package an excellent value.

MTBASIC CP/M BASIC Compiler, Softaid Inc., P.O. Box 2412, Columbia, Maryland 21045-1412. □

PROMAL

from Systems Management Associates

Programming language f Commodore 64

Review by Nick Sullivan

For most people, the Commodore 64 is a bilingual computer. For those who value a comfortable programming environment over performance, it offers the rather impoverished BASIC 2.0 dialect that, despite all drawbacks, has been the vehicle for some wonderful programs over the last few years. The 64 also makes life pretty easy for machine language programmers, who have a wide range of excellent tools at their disposal, and an unsurpassed wealth of documentation to light their way.

In between lies a large middle ground that has seen lots of contenders, but no clear winners. Ingenious programmers have contrived to adapt Pascal, C and Forth to the limited memory and sluggish disk speed of the 64. Several implementations of each of these languages are available, but none has caught on in a big way. Perhaps the compromises necessary to make them work in a relatively restricted environment have strained their integrity too far. The very impressive cartridge version of COMAL has fared somewhat better than the compiled languages, having been designed from the ground up to maximize the potential of the machine, but its growth has been inhibited by its dependence on the language firmware — you can't run a COMAL 2.00 program unless you have the cartridge.

In PROMAL, written by Bruce D. Carbrey, programmers now have the option of a compiled language designed specifically for three 8-bit microcomputers, the Commodore 64, the Apple IIc and the Apple IIe. Syntactically, PROMAL most closely resembles a stripped-down version of C, though programs are compiled in one pass in the manner of Pascal. Its best feature is its ability to maintain source code, object code, editor and compiler simultaneously in memory for short programs. Programs needing more elbow-room can be handled by compiling from and to disk and, if necessary, temporarily unloading the editor. In this way, the system is capable of generating substantial programs — up to 35K of combined code and data.

PROMAL's programming environment is a 'shell' — a sort of control centre from which you can invoke the editor or the compiler, and which also provides commands for disk operations, checking system status, and so on. Like the other parts of the system, the shell has been constructed with careful attention to detail. For instance, it is possible to retrieve the most recently issued command at a keystroke (**ctrl-b**); further repetitions of **ctrl-b** recall previous commands up to the limit of a 256-byte buffer. Users of the Amiga CLI (Command Line Interpreter) would sign away their limbs and vital organs for this facility. Some of the other features provided by the shell are online help, good command-line editing, and memory dump and fill commands in the manner of a machine language monitor.

Like the C-64's resident BASIC editor, the full-screen PROMAL editor is not intended as a general-purpose tool, but has instead been designed as an integral part of the total system. Most notably, it provides features that simplify the proper indentation of source programs, for indentation in PROMAL is syntactically meaningful, not just cosmetic. Specialized as it is, however, the editor is both powerful and straightforward, with facilities like search and replace and block move that are nearly indispensable when you are writing long programs.

Now that we've got into the editor, let's take a quick look at the language itself. The similarity to C is so strong that PROMAL might almost be recommended as an intermediate step for those who are approaching C from BASIC. Like C, a PROMAL program consists of functions that may or may not return a value, that can employ local or externally-declared variables, and that may call themselves recursively. PROMAL's manner of handling variable types is similar to C's, though the range of types is severely limited — PROMAL does not offer **struct** or **enum** types, for instance, and allows arrays of one dimension only. There is no double-precision option for real numbers, either, but PROMAL's six-byte floating point numbers are at least two digits more precise than the single-precision numbers in most other languages. Strings in PROMAL consist, as they do in C, of null-terminated byte arrays.

There are many further parallels. PROMAL offers convenient means for dealing with pointers and addresses, a welcome borrowing of one of C's most powerful features. Then there's the standard output statement, called **output** here rather than **printf**, but altered only superficially from the C original. Lastly — because you must be getting the idea by now — PROMAL makes use of a standard library of routines plus several special purpose libraries for things like floating-point operations, relative file handling, and RS 232 support. These are incorporated into your program with an **include** statement near the start of your code.

All the above notwithstanding, PROMAL source code and C source code *look* quite different. Specifically, PROMAL does without C's ubiquitous open and close braces to mark program structure, without many of C's parentheses, and without the semicolons that terminate C statements. PROMAL's lack of an explicit statement terminator is due to a limitation: each statement must occupy one and only one program line. This confines statements to 80 characters, a restriction that the horizontally-scrolling editor enforces.

The absence of special punctuation to mark program structures is more interesting, for PROMAL does support versions of the usual structured statements (**while** and so on). However, in PROMAL you group the statements within a structure by *indentation*, not punctuation. Since a logical scheme of indentation is favoured by programming theorists as an aid to clarity, which simplifies debugging and maintenance, PROMAL effectively kills two birds with one stone by making indentation a syntactic requirement.

PROMAL uses a one-pass compiler, like Pascal. This makes for speedier operation than a two-pass compiler (as in C) would allow, but it does have drawbacks for the programmer. A one-pass compiler requires that all references to variables, procedures and functions be backward rather than forward — the compiler needs to know what you're referring to before you actually refer to it. (There is a small exception to this rule, necessary to permit recursion, but it holds for all other situations.) This means, for example, that your mainline function must be the last to appear in your source code, and that your most general sub-functions must be defined after the specialized sub-sub-functions they may invoke. This calls for bottom-up programming, in defiance of those same theorists whose preference for indentation was noted above.

Advertisements for PROMAL lay great stress on the language's speed and compactness — qualities associated above all with machine language. It comes as something of a surprise, then, to find that PROMAL compiles to pseudo-code rather than machine code, and that some 12 kilobytes of run-time support are needed to underpin even a trivial program. This is not to say that the advertising claims are without foundation: PROMAL is undeniably much speedier than BASIC for many applications, and programs are very compact when you discount the 12K overhead. Still, it seems likely that some of the Pascal and C implementations for the 64 that *do* compile to ML would generate faster code than PROMAL in equivalent applications.

The PROMAL manual is perhaps the best I have seen for any C-64 language. It is clear, well-organized, accurate and comprehensive. It consists of an introductory tutorial, and separate reference sections for each of the major components of the system, including detailed descriptions of all library functions. Even those with no prior knowledge of a compiled language should have little difficulty in mastering PROMAL after a thorough reading of this excellent documentation.

The package comes in two flavours: one for programmers and one for developers. The primary difference between the two (apart from cost) is that the developer's version includes a facility for generating stand-alone programs, whereas programs written with the programmer's version can only be run under the PROMAL shell. This is barbaric. The manufacturer's additional effort in producing the developer's version must have been comparatively small; the difference in cost between the two versions is therefore nothing more than a price penalty one must pay in order to do serious work with the system.

PROMAL offers no built-in support for graphics and sound programming on the 64; nor are library functions for these purposes included in the package. Supplementary (looseleaf) documentation in the package I reviewed refers to a "hi-res windowing package" as an example of optional software available from the manufacturer (again at extra cost), but those who might wish to use sound, sprites or hi-res on their own will find no aid beyond a clunky, simple-minded video game (presented as an advanced sample program on the disk).

Despite these drawbacks, PROMAL is an ingenious and largely successful attempt to provide a compiled language congenial to the world's most popular computer. It is not likely ever to make the transition to the 16-bit world, where the advantages of C are much more compelling, so it does not have the strong argument of widespread portability in its favour. But for those who are frustrated by BASIC, yet intimidated by C or machine language, and for whom portability to other computer brands is not a concern, PROMAL offers a host of good features that make it worthy of serious consideration. □

Super Graphix Jr.

from Xetec

Printer interface for C-64 and VIC 20

Review by Ranjan Bose

This small black box will, for $78.00 (Cdn.), let your Commodore computer communicate with parallel printers.

Hookup is easy. The unit has three cables connected to it. The serial cable goes to the serial port on your computer or disk drive. A wire ending in an adapter goes to the cassette port to draw power. (A Datasette can still be used — it plugs into the adapter.) The parallel cable provided connects the unit to the printer.

Eight easily-accessible DIP switches let you adjust line feed, printer type, interface mode, device number and font. You can either use the regular dot matrix font provided by your printer, or use the interface's Epson-like correspondence quality font. The interface allows you to work in 1525 emulation mode (ignoring non-1525 codes), ASCII mode (converts PETSCII to ASCII), transparent mode (no translation, no modification) and a 'Super Graphix' mode, which is almost like all the other modes rolled into one. In this mode you can send 1525-compatible codes (graphics, Commodore graphic characters, reverse field characters) and also send non-Commodore codes to your printer (italics, emphasis, superscripts, high-density graphics, and so on). This mode also permits you to list the graphics and control characters in a program in four different ways: as they appear on screen, as mnemonics, as key-presses (**shift-c** would be listed as SC) or as ASCII codes.

You can access more interface functions by opening several channels with different secondary addresses for printing with upper case/graphics, with upper/lower case, in transparent mode, with or without line feed or in hex dump format. Any of these modes can be 'locked in' by adding 20 to the secondary address. You can also open a command channel to specify or change the several listing modes and to unlock the interface. I have written a program that prints text, Commodore graphic characters and quad density graphics all on the same line. This requires that I communicate with the printer in the emulation and transparent modes at the same time. From what I hear, most other interfaces are allergic to this kind of mixing.

The interface has a number of other attractive features. The DIP switches are 'active': should you want to change a setting, you do not have to power down. The quality of the graphic characters and reverse field characters found in the interface ROM is much superior to that available on Commodore printers. The accompanying manual is as well put together as the interface.

At present day prices for parallel printers and interfaces, it no longer makes sense to buy a Commodore printer and forego all the features and flexibility available on other printers. Xetec's Super Graphix Jr. is an excellent interface and I highly recommend it.

Super Graphix Jr. parallel printer interface, from Xetex Inc., 3010 Arnold Road, Salina, KS 67401. □

Elite
from Firebird
Space simulation game for Commodore 64

Review by Thomas Jones

The word 'elite' means 'the best', and it is an appropriate name for this new program for the Commodore 64.

Using a combination of high-resolution colour graphics for the cockpit and instruments, and three-dimensional wireframe graphics for the view of outer space through the windows, this simulation convincingly puts you into the hotseat of an interplanetary lone-wolf trader in the mid-thirtieth century. It is a ride to remember.

Unlike simple shoot-'em-ups, **Elite** is an interesting mix of several challenges. It delivers such a realistic simulation of navigation and docking in three-dimensional space that it could become the **Flight Simulator II** of spaceship simulators. The sensation of disorientation common to astronauts is also frequently felt when flying the **Elite** craft.

In addition to flying the ship and fighting, however, you must try to turn a profit in Wall Street fashion, trading goods to earn money to equip your basic Cobra trade vessel. Starting with a 100-credit stake, and a single pulse laser for defence, you must shrewdly assess the governments of many planets and the possible needs of their races.

Even if you develop a keen sense for profitable trade, **Elite** remains difficult, because many dangers await. Around many planets, and especially poor planets with dictatorships or in anarchy, pirates are ever-hungry for a rich cargo. The action is fast and deadly when you meet them, and you may find yourself crying out "What would Han Solo do now?" — it will take more than fast reactions to defeat many of these ships.

If you do shoot down a pirate, you may receive a bounty from GalCop, a big brother police force. GalCop sees everything you do on its super-radars, and knows you by your ship's radar signature. Not all ships you meet are pirates, however. If you attack a law-abiding ship, GalCop will put a price on your head. If this happens, on your next trip you will meet bounty-hunter ships, manned by excellent, combat-rated pilots and equiped with heavy armaments. Worse, GalCop itself is present in most systems to some degree, and will attack a criminal ship aggressively with fast combat Vipers when approached.

Assuming that you survive and keep your nose reasonably clean, you can amass money to further equip your ship in whatever way you deem fit. If, for example, you buy external scoops and a mining laser, you can go prospecting for asteroids. You sell the ore as well as trading goods, and you may occasionally find a few canisters of cargo in space to salvage. If you remain on the right side of the law, you can expand your cargo space, add an escape pod and better weapons, and trade on unstable planets that, while they attract unsavory pirates and rogues, pay premium prices for your cargoes. You can try trading in slaves, drugs, and guns if you can weather the wrath of GalCop when you try to dock at a civilized Corporate State or Democratic Multi-Government planet. As a last resort, if you can't make an honest living, you can even turn pirate yourself. Pirates destroy trade ships and scoop up their cargoes.

Your main navigational instrument is a remarkable 3-D radar scanner showing everything in your immediate area. You learn to use this to fly to the station, dock, and to engage (or flee) enemy ships. Other instruments to help you include indicators for speed, fuel level, shield power, cabin temperature, altitude above planet, laser temperature, energy bank charge, roll and pitch.

In the event you exhaust the trading possibilities in this galaxy, and if you can get up the price of an inter-galaxy Hyperdrive unit (sold only on high tech worlds), you can move on to the next of eight galaxies! I may never get out of this one.

Game options include a fast-load on/off, keyboard or joystick, enhanced planet detail on/off (there is a slight cost in speed for using this option), music on/off and more. You can save your current position to any formatted disk and restore it later. It retains all the positions saved, so you can return to any save point. Each save uses one block on the disk, and a unique code number is issued at each save, which you can send in to Firebird to enter a contest. More on this presently.

The program consumes about a third of the disk, and loads entirely into memory on the initial boot. It loads fairly quickly even without the fast-load option. I have had some problems with bad loads resulting in garbage on the screen, but it may be that this just reflects a glitch with my copy of the game.

Documentation for **Elite** is very complete and of excellent quality. It includes a sizable instruction manual, written in the style of the *USS Enterprise Operations Manual*. For added entertainment, you also get a science fiction novel based on the game! To help with the game-play, you are given a keyboard overlay, a chart of the types of ships, and a reference card to help you remember the commands.

The complex story line and high level of detail reflects a mature program with a following like that of the *Dungeons and Dragons* game. Indeed, **Elite** is but the latest version of a program nurtured and honed on the English BBC Home Computer. It is reputed to be the finest game ever created for this computer, and I believe it will be a classic for the C-64 as well.

Firebird is a London-based company, apparently distributing **Elite** under licence from AcornSoft (the BBC Home Computer is built by Acorn). When I ordered my copy of **Elite** from a dealer in Herts, England, I received a cassette with the European (PAL video system) C-64 version on it. This would run fine on PAL systems, but not on my American (NTSC video system) C-64, so I contacted Firebird for a replacement. I was mildly surprised at the promptness with which the Firebird staff obtained a USA-version disk from their United States subsidiary. They even sent a cassette with one of their new games on it to compensate me for any inconvenience! This is rare service.

If that isn't enough for you, though,

Firebird offers a monthly contest to find the best **Elite** pilots (remember that coded number at each save?)

If you can only afford one game this year, I suggest **Elite**!

Elite, $29.95 (US) from Firebird Licensees, Inc., P.O. Box 49, Ramsey, New Jersey 07446 U.S.A. □

Power Plan
from Abacus Software
Spreadsheet program with graphics for Commodore 64

Review by Dave Powell

My first impressions of this spreadsheet were not good. It swore at me in German when I tried to **Fast Load** it and, once loaded, flashed the borders and wailed at me whenever I made a mistake. During my first session with it, it crashed. However, I hadn't tried the graphics yet, so I persevered. I was coming at **Power Plan** having used a dozen spreadsheets previously. And it's different! Now that I'm used to it, it doesn't wail so much. (By the way, I never could recreate the situation that crashed it.)

I liked some features I hadn't seen elsewhere: showing *all* the formulae on the screen; good use of colour, including the ability to colour any cell, any colour; and a bar on the input line to show how much of my text would fit in the current column. I can change my mind about the cell I meant to use, while in the middle of entering data; and a single key will repeat the previous command, without a sidetrip through the menus.

I wasn't crazy about the manual. Some typos (and screen text, too) appear to be a product of the translation from German. ("Enter J for Yes"; and "Pie Cake" graph.) Another complaint about the documentation: it's too detailed! The training manual drove me to distraction by explaining every point to the tenth degree. But I have to say it's complete. **Power Plan** has too many options in some cases (*Do you want to change cell references in remaining formulae after deleting a column?* Yes, of course!), and too few in others (you can't choose the disk i.d. when you format a disk). My overall feeling is that it has slightly less spreadsheet function than its contemporaries, but certainly enough for most purposes short of small business. I found it harder to use, but most of that may be ingrained habit.

Power Plan's most distinctive feature, though, is its graphics.

When your data is calculated, choosing the graphics option leaves the data in memory, and loads new code to draw pretty things. These are presentation graphics — good for showing, more than using. There are no measurements on the axes for instance. A good set of options are provided: bar, pie, point, min-max (stock-price type) and points joined (straight lines). The first two also come in 3-D versions which adds flair. No titling is done for you, but a text editor allows titles, values, labels and whatever to be added. Best of all, up to eight graphs (any mix) can be put on one screen in your choice of sizes.

This is not a production tool, because the added text cannot be saved for use on a later graph. Partly because of this, getting a good-looking graph can be time consuming. Each change of data means a reloading of the spreadsheet code, then reloading the graphic part (two minutes each), and then redoing the labeling.

The graphics are better than any provided on any other C-64 spreadsheet I have seen, and don't appear to have increased the price. This might sway your decision. If you already have a fairly recent spreadsheet, this package might not give you a compelling reason to switch. If you don't, take a good look at **Power Plan.**

Power Plan 64, *from Abacus Software, P.O. Box 7211, Grand Rapids, Michigan 49510. Disk and manual, $39.95 US.* □

PR-1011 printer
from Roland
Dot matrix printer with Centronics port

Review by Ranjan Bose

Roland Canada has been marketing Panasonic printers for some time now, and the PR-1011 is the Panasonic KX-P1091, selling for $366.00 Cdn.

The 88 software commands that it supports are compatible with Epson codes, ensuring compatibility with most commercial software. The printer also supports special word processing commands for justification, centering, setting up of margins and so on. The draft quality output is pumped out at 100 cps (63 lines per minute) in pica, elite, semi-compressed, compressed and proportional pica pitches (10 to 17 cpi). Also available are print styles such as double width, emphasized, double strike, italics, superscripts and subscripts, underlining, 11 international character sets.

The character set is beautifully designed, especially if you use the near letter quality (NLQ) mode. You can switch this on either by printing software codes or by flipping a three position switch that selects between draft, NLQ and compressed modes. The NLQ prints at 20 cps and you can use any pitch in regular, double-width or emphasized style. The NLQ character set is very similar to that seen on typewriters and is very professional in appearance. It is the best NLQ printout that I have seen on a printer at anywhere near the price.

The printer supports several page-formatting commands, variable line-spacing, horizontal and vertical tabulations, and several 8-pin and 9-pin dot-addressable graphics modes ranging from single-density (60 dpi) to quad-density (240 dpi). You can also design and use up to 40 custom characters as well. The printer comes with a Centronics parallel interface and a 1K receive buffer (about 1/2 a page of text). A serial RS-232 interface and a 4K expansion buffer are available as options. Eight DIP switches are placed under the printhead track to control various printer parameters. Most Commodore-64 owners, however, will be using the switches on their interfaces, and need never bother with the printer DIP switches.

The printer has both friction and tractor feed, and permits trouble-free use of single sheets or fanfold paper up to ten inches wide. The printer is adequately dampened acoustically, and can be made quieter still by printing at half-speed. It uses a seamless nylon ribbon rated at 2 million characters. When the ribbon is exhausted, a built-in ink reservoir is activated to print another million characters. Most such ratings are somewhat exaggerated. Three million characters represents more than 1500 double-spaced pages of text! Also, the actual number of printed pages goes down with the use of graphics printing and double-pass printing (NLQ, et cetera).

The printer has a self-test and hex-dump feature. The manual is comprehensive and lucid. The bottom line? It's a beautiful, flexible, very affordable, feature-laden printer with the best NLQ in its price category — most highly recommended. □

Products Received

Presented by Astrid Kumas

The following products have been received by TPUG Magazine *in recent weeks. Please note that these descriptions are based on the manufacturers' own announcements, and are not the result of evaluation by* TPUG Magazine.

The Nutritionist

The Nutritionist from Nanosec Corporation, 4185 South 300 West, Ogden, Utah 84403. Order hotline: 1-800-NANOSEC, 1-800-626-6732. Price: $34.95 (US).

Weight control, nutrition and fitness seem to be everybody's concern, and as they remain the topic of the day, computer users will surely welcome a new product called **The Nutritionist**.

The Nutritionist, a menu-driven program for the C-64, has been designed as "a day to day nutritional guide" to help the user determine an individualized daily diet based on his/her physical and work related particulars. It can be used by people who would like to go on a diet to lose weight, or by those who simply want to improve their eating habits and maintain present weight. The manufacturer, Nanosec Corporation, claims that the program has taken two years of research and development, and is based on government research and the professional expertise of a company called Nutrition Lifestyles.

There are three options displayed on the main menu: *Nutrition Guide, Food List* and *Exchange List*. In the first option, *Nutrition Guide,* the user is required to enter specific information which will calculate his/her daily calorie needs as well as the intake values of important food elements like proteins, fats, carbohydrates, vitamins, trace elements, and so on. In this option, the user can also refer to the 'portioning' system which determines the number of portions for each category of food (dairy, vegetable, fruit, bread and starches, meat, fat, high carbohydrates) he/she can consume in a day.

Having established personal nutrition requirements, the user can procced to the two other options — *Food List* and *Exchange List*. *Food List* allows the user to determine the nutritional value of various food items. There are 898 food items, including some popular fast-food products, stored in the program. The nutritional value of each product entered by the user, as well as the sum of all listed for the day, appears on the screen, showing how close the user meets his or her nutritional goals, and if necessary, what modification in the diet should be made.

Exchange List presents several categories of food products, in special measured amounts, that contain about the same amount of calories, carbohydrates, fat and protein. By this means, foods within each category may be substituted for one another. This option should be used when planning daily menus.

Jungle Book

Jungle Book Reading from Fisher-Price Learning Software. P.O. Box 1327, Cambridge, Massachussetts 02238. Price: $24.95 (US).

Jungle Book Reading for the Commodore 64 is an educational program for improving reading comprehension skills. It is aimed at children seven to twelve years old. The program presents sixty short stories based on Rudyard Kipling's classic jungle adventures. Each story is followed by a series of reading comprehension questions. A special check/score system guides children back to the text for another reading. Reading passages and questions are similar to those encountered on standarized reading comprehension tests. But there is one important difference — they are not part of any test, they are part of the game which involves many favourite characters from Kipling's book.

Mowgli has to find Bagheera the panther, trapped in one of the many jungle caves. To accomplish that, he needs the child's help. The child must read five stories to Baloo the bear and answer all the comprehension questions correctly. Then after each story, following Baloo's commmand, he/she must guide Mowgli through the jungle to find and pass one of the five animals which block Mowgli's way to Bagheera.

Ernie's Big Splash

Ernie's Big Splash from CBS Software, One Fawcett Place, Greenwich, CT 06836. Price: $14.95 (US).

Ernie's Big Splash for Commodore 64 is a preschool learning activity designed to develop a sense of direction, as well as the ability to plan, predict, and solve problems. Perhaps this does not sound like much fun for children aged four to six. But it is.

Kids will enjoy meeting Ernie and other Sesame Street characters on their computer screen. Ernie is taking a bath, and would like to get his Rubber Duckie into the bathtub. Children can help Ernie by building a pathway that leads Rubber Duckie from his soap dish into Ernie's bathtub. The pathway can be created from separate building pieces, with doorways to control the movement of Rubber Duckie: up, down, left or right — but not necessarily toward the tub.

The disk contains three games: **Ernie's Challenge, Ernie's Fun Pal Challenge** and **Ernie's Super Challenge**. Each game introduces a new element to increase the challenge for children who have become familiar with the basic idea of building the pathway.

As children get more and more skilled at creating a passage for Ernie's Duckie, parents can encourage them to predict the number of pieces they will need to complete the pathway. They can also ask them to design pathways using the least number of pieces possible, or as many pieces as possible. □

A VIC 20 Correction

Those of you who have tried to use **P-Term.V12K** contained on VIC issue (V)TL may have noticed a problem. Disk users are sometimes told to "Press Play On Tape". The problem lies in line 70, which uses memory location 186 to ascertain which device, tape or disk, was used to load the main body of the program. The simplest solution is to change the **Peek** in that line to a simple '8' so that the line reads:

```
70 if peek(12288)<>169
  then load "term/vic.c2",8,1
```

Also, the **List-me** says that the secondary program is a patch. In reality, it is the machine language portion of the terminal program — mea culpa. We apologize for any inconveniences this error may have caused.

Richard Best
TPUG VIC 20 Librarian

Ask Someone Who Knows

If you enjoy **Jim Strasma's** many books, and his articles in this and other magazines, you'll be glad he also edits his own highly-acclaimed computer magazine, now in its sixth year of continuous publication. Written just for owners of Commodore's many computers, each **Midnite Software Gazette** contains hundreds of brief, honest reviews.

Midnite also features timely Commodore' news, hints and articles, all organized for instant reference, and never a wasted word. Whether you are just beginning or a long-time hobbyist, each issue will help you and your computer to work together effectively.

A six issue annual subscription is $23. To subscribe, or request a sample issue, just write:

MIDNITE SOFTWARE GAZETTE
P.O. Box 1747
Champaign, IL 61820

You'll be glad you did!

Tune your Computer into • MIDI •

If you own a Commodore, Atari ST, or Apple, then you may want to turn it on to the world of Music with a keyboard and interface. We stock the most popular MIDI interfaces (starting at $49.95) plus we have a limited offer on the CASIO **CZ101** Digital MIDI Synth. for only **$269** It's easy to turn your computer into a player piano* with the QRS Player Piano Roll Disks that feature music from 1900's to today (from Scott Joplin to Madonna). Interfaces with a 6 song music disk start as low as **$49.95** and the demo has Gershwin playing Rhapsody in Blue.

We have sequencer software, cables, CASIO and KORG keyboards, music disks, MIDI interfaces, plus more. Write or call for more information.

CALL - 201 - 838-9027
or write us at
Micro-W.
DISTRIBUTING INC
1342B Route 23 Butler N.J. 07405

DEALER INQUIRES INVITED
*REQUIRES MIDI KEYBOARD

COMSPEC

AUTHORIZED COMMODORE DEALER SINCE 1978.

HARDWARE • SOFTWARE • BOOKS • ACCESSORIES • SERVICE • RENTAL

866 Wilson Ave., Downsview, Ont. (between Dufferin & Keele) (416) 633-5605 or call our 24 hour B.B.S. 633-0185

KO-REC-TYPE®
RX, MX & FX RIBBONS
RED, BLUE, GREEN, BROWN
$13.95 EACH

Font Master II
WORDPROCESSING
With 30+ Fonts
including Hebrew & Greek
$69.95

COMPUTER DESKS
FROM
$89.95

KRAFT
JOYSTICK
MAZE MASTER
4 or 8-Way Selection
$16.95

MODEMS
NOW FROM
$49.95

PRECISION DISKS BY XIDEX
ss/dd ds/dd
$14.95 $17.95

10% DISCOUNT TO TPUG MEMBERS
ON SOFTWARE, BOOKS AND ACCESSORIES
(EXCLUDING SALES ITEMS)
PHONE ORDERS ACCEPTED
VISA/MASTERCARD/AMEX
(416) 633-5605

Bulletin Board

Computers in education

The Centre for Advanced Technology in Education has published the proceedings of the 1985 Summer Conference: *Extending the Human Mind: Computers in Education*. The 388-page book is divided into three sections containing 47 papers: **Curriculum Applications; Learning and Teaching;** and **Classroom and School Management.**

This book, and volumes containing the proceedings of previous conferences, can be ordered from ERIC Clearinghouse on Educational Management, Centre for Advanced Technology in Education, 1787 Agate Street, Eugene, Oregon, US 97403.

Also from Eugene, Oregon, comes the announcement of the formation of SIGCS — Special Interest Group for Computer Science — whose primary goal is to open communications between computer science teachers by providing a newsletter and helping to establish local groups.

The group has been formed under the aegis of the International Council for Computers in Education (ICCE), and is intending to explore and promote the teaching of computer science. The newsletter will cover such topics as classroom materials, ethical and social issues, and programming in a variety of languages. For further information contact SIGCS president Zab Warren, Phillips Academy, Andover, Maryland, US 01810.

Miami M.I.C.E.

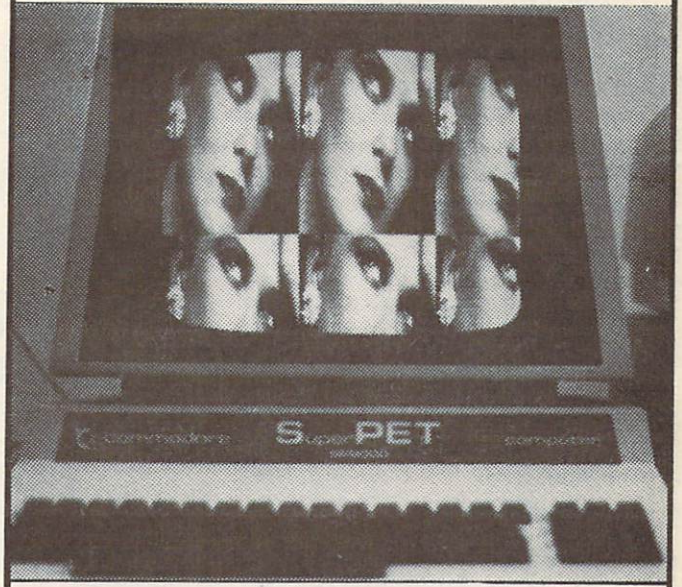
It has been brought to our attention that a new user group has been formed in the Sunshine State, called Miami Individuals with Commodore Equipment. Meetings are held on the third Friday of each month at the Florida International University Tamiami campus at 7:30 pm. The contact person is Ben C. Demby Jr., President, c/o 11110 Bird Road, Miami, Florida 33165, telephone (305) 221-7115.

Two other user groups have announced their formation: The MSD Disk Drive Information Exchange has been established to serve the needs of users of these products now that the manufacturer is no longer in business. Those who contribute information to the exchange will be entitled to receive: a list of software compatible with the drives, technical data, and facts about parts and service availability, as provided by other users. Further information about this non-profit organization can be obtained from Paul Eckler, 2705 Hulman St., Terre Haute, Indiana, US 47803.

The International Commodore Language Interest Group has been established to inform programmers about new tools and algorithms in programming languages. ICLIG publishes a newsletter, and maintains a public domain software library. Further information can be obtained from Kent Tegels, manager of ICLIG, at 1812 North I, Fremont, Nebraska, US 68025, or telephone (402) 721-4346.

Finally the Rochester FORTH Applications Conference is holding it's sixth conference at the University of Rochester, from the 11th of June till the 14th of June. The focus of this years conference is 'real-time artificial intelligence'. For more information contact Maria Gress at (716) 235-0168. □

INTRODUCING
THE HRT SUPER-RES
GRAPHICS BOARD



This is a graphics board which is universally compatible with all models of PET's. With Superpets (both 2 and 3 boards) It works equally well on both the 6502 and 6809 side as well as with OS-9.

Please send cheque or M.O. for $200.00 U.S./$ 5 .00 Cdn. plus $5 shipping and handling (Ontario residents please add 7% Prov. Sales Tax) To HIGH RES TECHNOLOGIES, 16 ENGLISHIVYWAY, TORONTO, ONTARIO M2H 3M4. Note: Please specify computer and disk drive model numbers.

(416) 497-6493

Classifieds

This space is for the ads of TPUG members. Wanted or for sale items only. Cost is 25 cents per word. No dealer ads accepted.

For Sale: IEEE interfaces for C-64. BuscardII, brand new, C64-link (416) 499-0797.

1526/802 Fonts: Triple Height Olde English, and Olde English Condensed; Double Height, Large Block and Single Height Olde English; Computer, and Greek fonts; Custom Letterhead program; and 5 character-high custom logo. Send $40.00 to Alphacom 15, Isabel Street, Petawawa, Ontario, Canada K8H 1Z1 (613) 687-2461.

advertisers' index

TPUG Magazine Distributors

***Dealers**: If you would like to carry **TPUG Magazine** in your store, you may order from any one of the following distributors:*

CANADA

Master Media, Oakville, Ont.	416-842-1555
Ingram Software, Concord, Ont.	416-665-0222
Compulit Distributors, Port Coquitlam, BC	604-464-1221

USA

Prairie News, Chicago, IL	312-384-5350
Levity Distributors, North Hollywood, CA	818-506-7958
Whole Life Distributors, Englewood, CO	303-761-2435
M-6 Distribution, Houston, TX	713-778-3002
The Homing Pigeon, Elgin, TX	512-276-7962
Northeast News Distributors, Kingston, NY	914-382-2000
Fred Bay News Co., Portland, OR	503-228-0251
Alonso Book & Periodical, Alexandria, VA	703-765-1211
Cornucopia Distribution, Seattle, WA	206-323-6247
Guild News, Atlanta, GA	404-252-4166
Micro-PACE, Champaign, IL	800-362-9653
Nelson News 4651 F Street, Omaha, NE	68127
Michianna News, Ft. Wayne, IN	219/484-0571
Total Circulation, South Hackensack, NJ	201/342-6334

TPUG Contacts

TPUG OFFICE 416/445-4524
TPUG BBS 416/273-6300
TPUG MEETINGS INFO 416/445-9040

Board of Directors

President	Chris Bennett	c/o 416/445-4524
Vice-President	Gerry Gold	416/225-8760
Vice-President	Carl Epstein	416/492-0222
Recording Sec.		
	David Bradley	c/o 416/445-4524
	Richard Bradley	c/o 416/445-4524
	Gary Croft	416/727-8795
	Mike Donegan	416/639-0329
	John Easton	416/251-1511
	Keith Falkner	416/481-0678
	Anne Gudz	c/o416/445-4524
General Manager	Bruce Hampson	416/445-4524

TPUG Magazine

Publisher	Bruce Hampson	416/445-4524
Editor	Nick Sullivan	416/445-4524
Assistant Editors	Tim Grantham	416/445-4524
	Adam Herst	416/445-4524
Production Manager	Astrid Kumas	416/445-4524
Ad Sales	John Matheson	416/445-4524

Meeting Co-ordinators

Brampton Chapter	Jackie Bingley	c/o 416/445-4524
C-64 Chapter	Keith Faulkner	416/481-0678
COMAL Chapter	Donald Dalley	416/742-3790
	Victor Gough	416/677-8840
Communications	Darrell Grainger	c/o 416/445-4524
Eastside Chapter	Nina Nanan	c/o 416/445-4524
Hardware Chapter	Frank Hutchings	c/o 416/445-4524
SuperPET Chapter	Gerry Gold	416/225-8760
VIC 20 Chapter	Anne Gudz	c/o 416/445-4524
Westside Chapter	John Easton	416/251-1511
	Al Farquharson	519/442-7000
Business Chapter		
New Users Chapter		
C-128 Chapter	Adam Herst	c/o 416/445-4524
Amiga Chapter	Mike Donegan	416/639-0329

Librarians

COMAL	Victor Gough	416/677-8840
PET	Mike Donegan	416/639-0329
SuperPET	Bill Dutfield	416/224-0642
VIC 20	Richard Best	c/o 416/445-4524
Commodore 64	Derick Campbell	416/492-9518
B-128	Paul Aitchison	c/o 416/445-4524
Amiga	Mike Donegan	416/639-0329
C-128	Adam Herst (CP/M)	c/o 416/445-4524
	James Kokkinen (C-128)	c/o 416/445-4524
MS/DOS	Colin Justason	c/o 416/445-4524

TPUG Bulletin Board

Sysop (voice, weekdays)	Sylvia Gallus	c/o 416/896-1446
Assistant Sysop	Steve Punter	c/o 416/896-1446

The Guide

A Monthly Publication For Commodore™ Owners

Commodore support with a twist . . . Personable and even humorous . . . Timely news . . . Helpful tutorials . . . On-going support for several languages: BASIC (including BASIC 7.0 as featured in the 128 PC), Machine Language, COMAL, and Pascal . . . Program Listings . . . *Honest* software reviews.

The Guide features some of the best computer humorists to be found.

- Introduce your "widow" to the Computer Widow's Compendium.
    - Tutorials and feature articles by the famous Mindy Skelton.
        - Featuring Shelly Roberts' "I'm Sorry . . . But I Don't Speak Hexidecimal." Discover why Shelly just may be the Andy Rooney of the computer world!

We feel we have assembled one of the most talented staff of writers in the Commodore world. Receive each month the most friendly and helpful Commodore publication available. Written by Commodore users who are writing to you, not down at you.

Limited offer — FREE With Each Subscription !

Subscribe *NOW* to capitalize on free software offer!

Receive up to $99.85 in *high quality* software!

With each year's subscription (or renewal) ordered, receive your choice of the software packages listed below, including the award winning educational games from Disney, or Omiterm Terminal written by our own Bob Richardson.

Subscription	Rate	Canadian *	Receive FREE
One Year	$18.00	$24.00 (U.S.)	Any software title listed below
Two Years	$35.00	$47.00 (U.S.)	Any two of the titles below
Three Years	$48.00	$64.00 (U.S.)	All three of the titles below

* Note: Canadian (and other foreign) subscribers must add $2 (U.S.) per program title to cover shipping and handling. No handling charge for U.S. subscribers. Canadian rates listed in U.S. dollars — send U.S. funds only, please.

Donald Duck's Playground

CES Software Showcase Award Winner! — Disney animation at its best! Children play four games to "earn" money to buy playground equipment. Builds money handling skills. Superb graphics. A bestseller! $39.95 retail value.

The Guide
3808 S.E. Licyntra Ct.
Portland, OR 97222

Winnie The Pooh In The Hundred Acre Wood

— Players explore the Hundred Acre Wood to find lost articles like Ow'ls books, Pooh's honey pot and Eeyore's tail, and return them to their rightful owners. Cheery music from the Disney movie caps off this computer rendition of the beloved classic. $39.95 retail value.

Omiterm Terminal Program

— Written by The Guide's own Bob Richardson. Fully supports the new 1660 Modem 300! (The *ONLY* commercial terminal package that currently fully supports the 1660!!!

- **Modem controls accessible from the keyboard**
- **Punter protocol — upload & download — 300/1200 baud**
- **Ten programmable function keys**
- **15 number phone directory**
- **20k receive buffer**
- **Tone or pulse dialing**
- **Auto dial/re-dial**
- **Half/full duplex**

A $19.95 retail value.

Don't delay — mail today! Supplies are limited.

Name____________________

Address____________________

City, State & Zip____________________

Enter My Subscription for:

☐ **1 Year** ☐ **2 Years** ☐ **3 Years**

Please Check One: ☐ **New** ☐ **Renewal**

You are entitled to one FREE software package, as described above, for each year's subscription purchased. Please list your second choice, as supplies are limited on the Disney titles! Please allow six to eight weeks for delivery.

My first choice is:____________________

____

My second choice is:____________________

____

Software designed for AMIGA.

Lattice C Compiler ***$149.95***

With more than 30,000 users worldwide, Lattice C Compilers set the industry standard for MS-DOS software development. Lattice C gives you all you need for development of programs on the AMIGA. Lattice C is a full implementation of Kernighan and Ritchie with the ANSI C extensions and many additional features.

AMIGA C Cross Compiler ***$250.00***

Allows AMIGA development on your MS-DOS system. Price includes the above product.

Lattice Screen Editor (LSE™) ***$100.00***

Designed as a programmer's editor, *Lattice Screen Editor (LSE)* is fast, flexible and easy to learn. *LSE's* multi-window environment provides all the editor functions you need including block moves, pattern searches and "cut and paste." In addition, *LSE* offers special features for programmers such as an error tracking mode and three Assembly Language input modes. You can also create macros or customize keystrokes, menus, and prompts to your style and preferences.

Lattice dBC III Library™ ***$150.00***

The *dBC III library* lets you create, access and update files that are compatible with Ashton-Tate's dBASE system. *dBC III's* C functions let you extend existing dBASE applications or allow your users to process their data using *dBC III* or dBASE III.

Lattice Make Utility (LMK™) ***$125.00***

An automated product generation utility compatible with UNIX Make, *Lattice Make Utility (LMK)* lets you rebuild complex programs with a single command. Once you specify the relationships of the various pieces of your system in a dependency file, *LMK* automatically rebuilds your system the same way every time, and only compiles program files that have changed. But *LMK* is not limited to updating programs. You can use *LMK* to update documentation or perform **any** executable command!

Lattice Text Utilities™ ***$75.00***

Lattice Text Utilities (LTU) consists of eight software tools to help you manage your text files. GREP searches files for the specified pattern. DIFF compares two files and lists their differences. EXTRACT creates a list of file names to be extracted from the current directory. BUILD creates batch files from a previously generated file name list. WC displays the number of characters and optionally the checksum of a specified file. ED is a line editor which can utilize output from other *LTU* software in an automated batch mode. SPLAT searches files for a specified character string and replaces every occurrence with a specified string. And FILES lists, copies, erases or removes files or entire directory structures which meet the specified conditions.

Lattice Unicalc® Spreadsheet ***$79.95***

Unicalc is a simple-to-operate program that turns your AMIGA computer into an electronic spreadsheet. Using *Unicalc* you can easily create sales reports, expense accounts, balance sheets, or any other reports you had to do manually.

Unicalc offers the versatility you've come to expect from business software, plus the speed and processing power of the AMIGA.

• 8192 row by 256 column processing area • Comprehensive context-sensitive help screens • Cells can contain numeric, algebraic formulas and titles • Foreign language customization for all prompts and messages • Complete library of algebraic and conditional functions • Dual window capabilities • Floating point and scientific notation available • Complete load, save and print capabilities • Unique customization capability for your every application • Full compatibility with other leading spreadsheets.

Lattice MacLibrary™ ***$100.00***

The *Lattice MacLibrary™* is a collection of more than sixty C functions which allow you to quickly and efficiently take advantage of the powerful capabilities of the AMIGA.

Even if your knowledge of the AMIGA is limited, *MacLibrary* can ease your job of implementing screens, windows and gadgets by utilizing the functions, examples and sample programs included with the package.

Other *MacLibrary* routines are functionally compatible with the most widely used Apple® Macintosh™ Quickdraw Routines™, Standard File Package and Toolbox Utility Routines enabling you to rapidly convert your Macintosh programs to run on the AMIGA.

Panel™ ***$195.00***

Panel will help you write your screen programs and layer your screen designs with up to ten overlapping images. *Panel's* screen layouts can be assigned to individual windows and may be dynamically loaded from files or compiled into a program. *Panel* will output C source for including in your applications. A monitor and keyboard utility is also included to allow you to customize your applications for other systems.

With Lattice products you get *Lattice Service* including telephone support, notice of new products and enhancements and a 30-day money-back guarantee. Corporate license agreements available.

Lattice

Lattice, Incorporated
Post Office Box 3072
Glen Ellyn, Illinois 60138
(312) 858-7950 TWX 910-291-2190

INTERNATIONAL SALES OFFICES:
Benelux: Ines Datacom (32) 27205161 England: Roundhill. (0672)54675
Japan: Lifeboat Inc. (03) 293-4711 France: SFL (1) 46-66-11-55
Germany: (49) 7841/4500